Teaching Object-Orientation with Smartphones as Digital CRC Cards

Rainer Lutz, Sascha Schäfer, and Stephan Diehl
University of Trier, Germany
{*lutzr, s4sascha, diehl*}*@uni-trier.de*

Abstract

Object-orientation is one of the essential parts of every software engineering course. However, according to literature, it often lacks the following: First, modeling on a conceptual level independent from a particular programming language is often neglected. Moreover, the actual process of designing or implementing a piece of software seems to be less important than providing an optimal solution. Finally, students often follow the slides of the teacher passively without thinking about or even questioning the material. But, in order to actively apply a proper form of object-orientation, it is essential that one understands the concepts of this programming paradigm and is able to deal with problems and communicate ideas rather than handling a specific programming language. In this paper we introduce CREWSpace—a tool for co-located collaboration that implements a digital version of the CRC method. To this end, students actively participate in so called CRC sessions, which fosters communication and problem solving skills. In particular, they design a software system on a conceptual level and use our tool to analyze and adjust the proposed design through role play. Moreover, CREWSpace records these analyses for a later replay such that the students are able to reflect on their decisions.

1. Introduction

Teaching object-oriented design and programming is an essential part of every software engineering course. Although over the years different approaches for teaching object-orientation have been proposed and discussed, there also seems to be room for improvement. In literature a lack of conceptual modeling, which focuses on the structure and concepts of object-orientation instead of a specific programming language, is often described. Furthermore, not only the result, e.g., a snippet of the source code or a model of the software system, is important, but also the process of creating this artifact [2, 6, 11, 21]. Independently from software engineering education, it is argued that students should also apply active learning techniques instead of only passively following the slides of the teacher without thinking about or even questioning the material [10].

One of the first approaches for teaching object-oriented analysis and design is the so called CRC[1] method introduced by Beck and Cunningham at OOPSLA in 1989 [1]. In their paper it is explained how a group of students can use a simple set of index cards and a few pens to create an object-oriented model of a software system. Therefore, each index card describes a class of the system and records its responsibilities as well as collaborating classes. When the students have identified an initial set of classes, role play is used to simulate the software system and to analyze its design. Here, each student acts for a certain number of classes and describes its activities during

[1] CRC stands for Class-Responsibilities-Collaborators

978-1-4673-5140-9/13 © 2013 IEEE CSEE&T 2013, San Francisco, CA, USA

previously identified scenarios. We argue that the CRC method is a decent technique to learn object-orientation on a conceptual level because it represents a rather simple modeling approach, i.e., no complex notations are needed, and moreover, it allows to focus on the central classes of a software system. Furthermore, it incorporates collaborative learning, which encourages students to share and discuss ideas and opinions [19].

Nevertheless, the traditional CRC method has some drawbacks in the context of education. For instance, during role play, which can be considered as the actual design process, the group has to remember the particular classes (and their order) that were required to simulate the software system in a specific scenario. Thus, it quickly becomes non-trivial to recall how the current state was reached or even reflect on the achieved results. However, this stands in contrast to the following statement from Collins and Brown:

> "The recording and replaying of the processes people use to perform tasks such as reading, writing, and problem solving, has the capability to make these processes objects of reflection, annotation, and communication. Using imitation, replay, abstracted replay, and reification, student's can begin to think about, talk about, and experiment with their learning and problem-solving processes in a way not previously possible." [8]

Another drawback is that there might be either too dominant students, which try to carry through their own ideas, or too quiet people that simply try to stay out of the role playing sessions. Finally, considering the outcome of a CRC session, one can easily see that handwritten paper-based CRC cards are impractical when it comes to transfer this information to subsequent steps.

With CREWSpace we want to remedy the disadvantages discussed above and, moreover, introduce modern technologies into the process of learning object-oriented software design. In summary, a tool that supports digital CRC sessions such as CREWSpace should:

- Allow students to replay previous analyses of the software system in order to reflect on its design and probably modify it to prevent design flaws.
- Somehow handle too dominating or too quiet group members to allow a fair collaboration between all students.
- Provide a possibility to preserve achieved results for subsequent steps.

2. Teaching Object-Oriented Design with Digital CRC Cards

In this section we introduce CRC sessions with CREWSpace. In order to use our prototype a single computer, preferably connected to a projector, and a mobile device for each student, i.e., smartphones or tablet computers running an Android operating system, should be available. Once the students entered the IP address of the computer into their mobile devices, these are able to connect to the main application of CREWSpace shown in the center of Figure 1. This Java program provides a shared workspace for all students and displays the digital CRC cards that will be created throughout the group session. Furthermore, the students can trigger a new role play in order to analyze the current software design, review a previous one, or even use simple UML class diagramming to enrich their design with additional information. To this end, the mobile devices serve as controls for the main application. That is, a colored cursor is assigned to each student and can be moved across the screen by using the mobile device as a touch pad similar to those integrated into common laptops. They also serve as private workspaces where CRC cards can be created or modified and additional information may be accessed.

2.1. Digital CRC Sessions

Similar to the traditional CRC method, a group of students designs and analyzes a software model with CREWSpace in three subsequent phases: First, use cases as introduced by Jacobson [13] and respective scenarios are identified[2]. Then, the group creates an initial set of CRC cards, which serve as a starting point for the third phase where they apply role play in order to analyze and enhance the software system based on the previously identified use cases.

Next, we describe a typical situation in which CREWSpace is used by a group of students to design a software system. An example is the system shown in Figure 1, which is an excerpt of a simple ATM available at [4]. We assume that the students were introduced into the metaphors and controls of CREWSpace and have already identified all important use cases along with particular scenarios in an earlier exercise. Alternatively, this information can also be provided by the supervisor. If stored in the main application, it is also accessible via each mobile device.

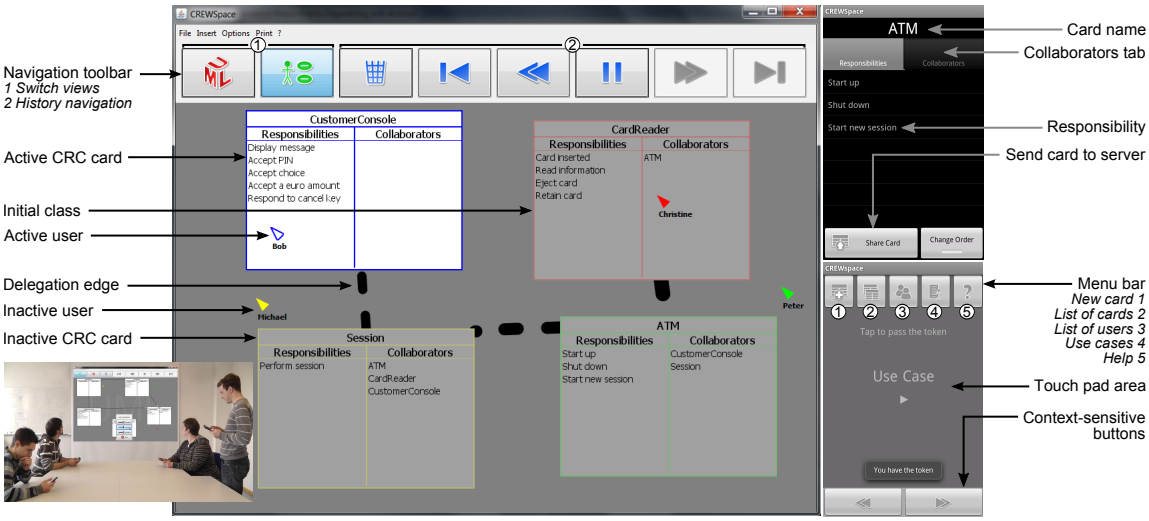

Figure 1. Main application of CREWSpace in Use Case View (center). Edit dialog and main screen of the mobile application (right). Example CRC session (bottom left).

Either way, the students should discuss their particular requirements as well as the use cases with the according scenarios and develop a general strategy. This done, the students start with identifying an initial set of classes and create new CRC cards for each of them. Therefore, they use their mobile devices to enter a name as well as obvious responsibilities and collaborating classes. Of these, responsibilities cover both the data a class is able to access and the tasks it can perform. For instance, Figure 1 shows the edit dialog for the ATM class in the top right corner. Like in the traditional settings students are able to create new cards simultaneously and post them to the shared workspace. There, all cards can be rearranged on screen and distributed over the group members by using the touch pad area depicted in the bottom right corner of Figure 1. Card ownership is indicated by a colored border, which matches the particular cursor color. For the next phase we suggest that each student should own not more than two cards because we observed that a smaller set of initial CRC cards made role play more accessible at first.

[2]Use cases comprise functional requirements and capture the externally visible behavior of a system from a user's perspective; scenarios describe a possible execution of or a path through a use case

Once all cards are distributed (in Fig. 1 only a single one per student), the group can begin with the role play by pressing the play/pause button on the navigation toolbar. As a consequence, all CRC cards are grayed out and the tool asks the students to select an initial class. At this point, the group has to decide which scenario they would like to analyze and what particular class they have to start with. The person who triggered the play button selects this card such that it becomes active which is indicated by rendering it fully opaque (cf. Fig. 1). Now the owner of the card comes into play. She explains which responsibilities her class has to fulfill according to the chosen scenario. If her class can achieve them without collaborating with other classes, she simply confirms this by clicking at her card. In case the class has to collaborate with another one, she can delegate the task to the particular class by clicking at it. Hence, a dotted delegation edge is shown on the shared screen (cf. Fig. 1); floating from the caller to the callee. Furthermore, the collaborator becomes the active class and is shown fully opaque. However, there might not exist a collaborator that contains the particular responsibility the group was searching for. In that case the students may choose between two options: either they add the missing responsibility to an already existing or to a newly created card. Afterwards, the group continues with this procedure until the scenario is completed. During a scenario play through, CREWSpace records a call history such that the students are able to revisit earlier stages of the role play, reflect on the design, and possibly change their model. Hence, the navigation toolbar provides buttons to browse the recorded history (cf. Fig. 1). When all use case scenarios have been covered, a complete model of the software system is available.

2.2. Additional Features

In an optional fourth phase the students can use CREWSpace to create a rudimentary UML class diagram. Therefore, they switch to a UML view where they are able to link CRC cards using aggregation, inheritance, and association edges and afterwards export their diagram for a further use with common UML tools.

During a CRC session it may occur that a student is too dominant. For instance, he possibly tries to force the teammates on his own ideas or simply wants explain the responsibilities of classes he is not responsible for. Thus, we included a simple feature: Students may vote for teammates that seem to be too dominant. If a certain amount of students share that opinion, the respective team member receives a message on his mobile device, which suggests to him to let other's have their say. For more information about the features of CREWSpace please consider our recent work [14].

3. Evaluation

CREWSpace was used during two software engineering courses at the University of Trier as well as in a recent lab session at the University of Antwerp. We collected feedback from both the students that worked with our tool and their supervisors. In particular, we observed four teams, in total 16 students, modeling predefined software systems. Of these, four people had applied the CRC method before—either with our prototype or in a traditional session. Table 1 shows the characteristics of each group.

Each of the team members had access to an Android-enabled touch device with preinstalled mobile application. In addition, either all requirements documents including use cases and diagrams about the system were provided or the students acquired this information during an earlier exercise, i.e., these groups were already familiar with software system. Furthermore, all students got a basic introduction into the traditional CRC method.

After a short briefing and a warm-up phase to learn the controls and features of CREWSpace, the teams were asked to model the software system. In order to observe which features of our tool

Table 1. Group characteristics

	Group 1	Group 2	Group 3	Group 4
Students	Graduate	Undergraduate	Undergraduate	Undergraduate
Group size (prior experience)	4(4)	4(0)	3(0)	5(0)
Time spend for modeling	44 min	40 min	37 min	47 min
System to model	online store	light control system	light control system	ATM

were used, we logged all activities including timestamps and user identifiers. Additionally, we got the permission of the first group to videotape their session for later analysis. After each session we collected feedback from all the students in short group discussions or in written form.

3.1. Observations

A lot of interesting observations were made during the lab sessions, which are summarized in this subsection. Although, at a first glance, all groups had somehow individual ways of tackling their task, we could also identify similar patterns. For instance, the alternation between modifications to CRC cards and their analysis via role play. In particular, some groups (e.g. Group 1) preferred to do only a few modifications and directly applied role play in order to check and discuss if these modifications are reasonable. In contrast, Group 4 manipulated their software model for a longer time before they actually validated it via role play. Obviously, there also exist compound strategies, i.e., some students applied longer phases at the beginning to yield a basic design and as soon as the model evolved, editing and role playing phases became shorter.

Another interesting observation is how the students used the history navigation feature. Although we explained to them that it is possible to go back to previous steps of a role play, the students were free to use this feature in whatever way they could imagine. Most of the groups navigated through the recorded history in order to replay a particular part of a previous role play to investigate their recent modifications to the software model (e.g. new classes, responsibilities, or collaborators) or even explore alternative solutions. In addition, Group 1 used the history navigation for a final validation of a particular use case scenario in order to reflect on the changes that were made to the software model. To this end, one student took the lead, went to the beginning of the recorded role play, and browsed once again through the entire scenario. At the same time, he explained what, in his opinion, the particular classes are responsible for and how they are collaborating with other ones. In case he was not sure about a responsibility, he simply asked the respective owner. The remaining students observed this process and intervened as soon as they spotted possible design flaws.

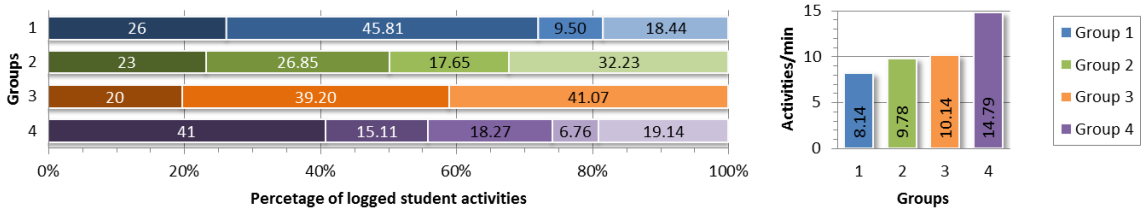

Figure 2. Distribution of student activities and comparison of group activities.

As CREWSpace is able to log all interactions with both the mobile devices and the main application, we used this information to explore how active the students were during the lab sessions. Figure 2 shows the percentage of logged activities for each student in the left diagram, which, in

general, indicates that all students participated in the modeling session. However, there are two students that participated more than others according to the log files; the second member of Group 1 (A) and the first member of Group 4 (B). While this effect can be easily explained for student A, he extensively applied the history navigation feature for the final validation of each scenario (see previous paragraph), Student B seems to explore the interface and features of CREWSpace a lot, sometimes for some unknown reason. Moreover, the left diagram also shows that each group had a single less active member. However, during role play such students are somehow forced to explain what their own classes are responsible for and thus have to participate in the discussion because CREWSpace does not allow other members to proceed with the analysis. We also think that it is more likely that a student grabs a paper card of a nearby teammate than taking the mobile phone to proceed with the role play in her position. The right bar chart depicts the activities per minute for each group. While the first three teams show a similar amount of activities per minute, the members of the last group were much more active because they seemed to like exploring the interface.

In Subsection 2.2 we explained that CREWSpace can also be used to model and later on export rudimentary UML class diagrams. Although it was intended to use this feature after a complete model of a software system has been designed, some groups also applied UML modeling in combination with their analysis to enrich the software model with additional information, in particular, to augment it with inheritance, association, or aggregation relationships.

In summary, our observations show that CREWSpace can be used in different ways and the group is not restricted to a certain workflow. Furthermore, it allows the students to collaboratively reflect on their design which helps sharpening their understanding of problem solving in the context of object-orientation. In addition, Figure 2 suggests that each student was interacting with our prototype and thus involved into the design of the software system.

3.2. Student Feedback

This subsection summarizes the students' impressions of our prototype, which were collected after each particular CRC session. In general, the students liked working with our prototype and saw many advantages over the traditional variant. Their feedback was mostly positive, e.g., people made comments like "motivating", "clear presentation", or "interesting variant". However, the students also made suggestions for improvement, which we discuss at the end of this subsection.

The quality of any interactive tool stands and falls with its usability. Especially graphical user interfaces of tools for education should be as intuitive and self-explanatory as possible. Hence, the students were asked to report on the usability of CREWSpace. Although there were no major complains about the main application, which follows a common window-based user interface, for the mobile application there was no overall consensus on the user interface and the according touch gestures. While most students understood the controls directly, others stated that it took some time to familiarize with them. A reason for this might be the fact that various mobile devices and even applications provide different user interfaces and touch gestures. Either way, the students seemed to be motivated to help each other. A few students with smaller mobile devices found it somehow tedious to edit CRC cards using an on-screen keyboard and would have preferred writing their cards by hand. However, once the students got used to the controls, they appreciated to use the mobile devices as private workspaces and that they were able to modify different CRC cards simultaneously.

Furthermore, the students were asked to compare the traditional CRC method with the digital variant. Almost all students appreciated that during role play our prototype is able to show the current state and, moreover, how it was reached as this turned out to be a weakness of the traditional method. Also, the ability to save or export their work for subsequent exercises was mentioned as an

important advantage. In addition, one particular student found that a digital variant has much more potential for automation. For instance, he suggested to add collaborating classes automatically during role play. As mentioned by most of the students, a major drawback of our approach is setting up the hardware and connecting the mobile devices to the main application. However, this procedure is actually quite simple and quick, but compared to the traditional method still slower.

Inspired by the interactive lab sessions, most of the students come up with ideas for improvement. For instance, they suggested to include a tutorial video that explains the functions of CREWSpace in case no instructor is available. This way students could also use our tool to design software systems during self-organized sessions (cf. Subsec. 3.4). Some students mentioned that the screen size can quickly become a problem for larger systems. Although CRC cards are not intended to model every single detail of a system, i.e., the number of classes should be manageable most of the time, we agreed with the students and recently integrated an option to scale all cards.

3.3. Supervisor Feedback

Besides feedback from the students we also asked their supervisors to assess the particular lab sessions based on a short questionnaire. The following paragraphs summarize their answers.

Did you have problems settings up the lab session? If so, please describe them. Although CREWSpace is actually quite flexible when connecting the mobile devices to the main application, in some lab sessions the supervisors reported network problems. Mostly due to restrictions of the particular university network. However, these problems could be solved quickly. For example, by setting up an ad-hoc network with one of the mobile phones or using a different router.

Please tell something about the motivation of the group. Was it hard to find a group of students that wanted to participate? What did the other students do? Finding students who wanted to participate in a lab session using CREWSpace was less of a problem as testing a new tool seems to be very motivating. However, as the supervisors were not able to provide Android devices for all students, most of them had to bring their own devices. Obviously, this reduces the number of possible participants drastically. Nevertheless, the supervisors were able to form at least one group per course while the rest of the students applied the traditional variant using pen and paper.

Please describe the group dynamics. How did the students interact with each other and with CREWSpace? Most student were very curious and started already exploring the functions of CREWSpace during the introduction. They created new classes, moved them around, and investigated the client application which is also a part of the learning experience. Moreover, it seems to be advantageous when the students already know each other. During the actual CRC session some supervisors observed an alternation between group discussions and individual work. In particular, subsequent steps, e.g. introducing new classes, responsibilities, or collaborators, were discussed in the group, distributed among several students who applied them, and validated through role play.

Was there a dominating student? If so, how did he dominate the group? In general, the supervisors did not report any prominent cases of dominating people meaning that all students participated in the group discussions. Needless to say that there exists a certain fluctuation as some students may simply contribute more than others. This is also a reason why some supervisors observed a quiet student per group that did not talk as much as the others (also cf. Fig. 2). However, our tool somehow forces those students to actively participate in the role play as soon as they have to describe what their classes are responsible for.

Did you play a passive or an active role? Did you interfere if something was obviously wrong or modeled in a bad way? If yes, how did you interfere? All supervisors reported to be active during the introduction of CREWSpace but afterwards played a passive role. If intervention was required, they tried to make the students aware of the problem rather then telling the particular solution.

3.4. Learning Scenarios

During the lab sessions with CREWSpace we discovered different learning scenarios, which we want to introduce in the following.

One way CREWSpace can be used for teaching object-oriented design is to integrate one or more CRC sessions into a software engineering course. Instead of providing an arbitrary example for such sessions, we found that a complete software project throughout the whole course motivates the students much better. This means that the students go through the phases of designing a given software system, i.e., they look for requirements, create use cases and according scenarios, and finally develop a model of the system. Most of these tasks are done as homework assignment. Assuming that the students have prepared requirements as well as a set of use cases and hence are familiar with the software system, the CRC method is used to design a basic model of the software system. Before the actual lab session, all students should get a basic introduction into the traditional CRC method. Students that are willing to try out CREWSpace (and also have an Android device available) are split into groups of three to six participants. The remaining people apply the traditional CRC method with index cards and pens. This allows the teacher to compare both variants with the students later on. Digital CRC sessions are conducted as described in Subsection 2.1. While the supervisor may be open for questions, we suggest that he should only play a passive role with as less intervention as possible. At the end of the CRC session, the students may also save or export their finished design for the use in further exercises.

Alternatively, the supervisor could actively join the group of students and guide them through the process of designing a model of the software system. Similar to the application of Supplemental Instruction where upper grade students somehow coach younger ones [21]. Furthermore, so called Coding Dojos [18], where a group of people meet in order to train their programming skills when developing small projects, might be interesting to investigate. For instance, during our own Coding Dojos we observed that we were often forced to formulate the problem in an object-oriented way, mostly on a conceptual level, before we started programming. Here, CREWSpace could be an ideal tool to create a model of the software in a collaborative CRC session.

3.5. Threats to Validity

In this paper we qualitatively evaluated CREWSpace in the context of software engineering eduction. This is, we observed how students interact with each other and with our tool rather then quantitatively measure or compare the effectiveness of the learning process. We collected feedback from both the students and their particular supervisors such that we were able to investigate opinions from different points of views. Both the fact that the lab sessions were limited to a certain time frame, i.e., none of the teams was able to model the entire system, and the small number participants may limit the validity of our results. In longer experiments the groups would have had more time to better adopt the metaphors and the controls of our prototype.

4. Related Work

Of course we are not the first that adopted the CRC method for software engineering education. Other researchers have also investigated this approach and integrated it with their idea of teaching object-orientated design and programming. For instance, Börstler et al. [6] and in a subsequent research paper Westin and Nordström [21] briefly explain how the CRC method can supplement their course design. However, as already discussed in Section 1, researchers also identified weaknesses of the CRC method and suggested possible solutions or adjustments [3, 5, 20]. For example,

Börstler [5] observed that CRC cards are used as object surrogates during role play although they actually model classes. He proposes to use special object cards as instances of a CRC card. CREWSpace could easily be extended to create such object cards automatically during role play.

There are quite a number of papers that describe approaches to object-oriented analysis or design, for instance, Knight [9] and SUMLOW [7] implement co-located collaboration for UML modeling on an electronic whiteboard or Calico [15] allows users to collaboratively draw and discuss early design sketches. However, to the best of our knowledge, only a few provide ideas on how these approaches can be used for teaching object-orientation. Next, we briefly describe CRC tools that were specifically designed for education purposes.

EasyCRC [16] was developed to teach students object-oriented design. In comparison to other tools, it assists students all the way from the beginning of a CRC session where only a textual description is available to the usage of sequence diagrams for role playing in order to get a complete set of CRC cards. CRC Design Assistant [17] is another tool that supports students during the design process. Focused on data management it stores CRC models in a database and provides features for effectively creating and maintaining CRC cards. Additionally, ECoDE [12] provides features for source code generation. Therefore, classes and scenarios are identified, responsibilities and collaborators assigned, and finally methods and attributes created. While these systems are more complete, in a sense that they support and maintain more or less the entire designing process, our tool concentrates on the collaborative part of the CRC method, involves all students in group discussions, and thus fosters communication and negotiations skills.

5. Conclusions

In this paper we introduced CREWSpace, a tool that implements a digital variant of the CRC method. We showed how co-located collaborative CRC sessions with our tool are conducted and explained how students can replay previous analyses to reflect on their design. Moreover, CREWSpace allows to preserve the achieved results for later steps. Our evaluation shows that these features were actually used in different ways and that the students perceived our tool positively. Investigating the logged activities as well as the teacher questionnaire showed that dominant and quiet students cannot be avoided, but, nevertheless, decently handled through the card ownership during role play. The teacher questionnaire provided further information about technical problems and the group dynamics from the supervisors' point of view. Finally, we discussed how CREWSpace can be integrated into different learning scenarios.

Although CREWSpace only supports collaborative CRC sessions, we think that many of the concepts presented in this paper can be transferred to collaborative analyses of other kinds of software models.

Acknowledgment

The authors would like to thank Quinten Soetens for using CREWSpace during a lab session of the software engineering course at the University of Antwerp.

References

[1] K. Beck and W. Cunningham. A Laboratory for Teaching Object-Oriented Thinking. In *Proc. of Conference on Object-Oriented Programming Systems, Languages and Applications, New Orleans, LA, USA, October 2-6*, pages 1–6. ACM, 1989.

[2] J. Bennedsen and M. E. Caspersen. Teaching Object-Oriented Programming - Towards Teaching a Systematic Programming Process. In *Proc. of the Workshop on Pedagogies and Tools for the Teaching and Learning of Object-Oriented Concepts, European Conference on Object-Oriented Programming, Oslo, Norway, June 14-18*. Springer, 2004.

[3] R. Biddle, J. Noble, and E. Tempero. Reflections on crc cards and oo design. In *Proc. of International Conference on Tools Pacific: Objects for internet, mobile and embedded applications, Sydney, Australia, January*, pages 201–205. ACM, 2002.

[4] R. C. Bjork. An Example of Object-Oriented Design: An ATM. http://www.math-cs.gordon.edu/courses/cs211/ATMExample/ (accessed in January 2013).

[5] J. Börstler. Improving CRC-card role-play with role-play diagrams. In *Companion to the Conference on Object-Oriented Programming, Systems, Languages, and Applications, San Diego, CA, USA, October 16-20*, pages 356–364. ACM, 2005.

[6] J. Börstler, T. Johansson, and M. Nordström. Teaching OO Concepts—A Case Study Using CRC-cards and BlueJ. In *Proc. of the Frontiers in Education Conference, 2002, Boston, MA, USA, November 6-9*, pages T2G – 1–6. IEEE, 2002.

[7] Q. Chen, J. G. Hosking, and J. C. Grundy. An E-whiteboard Application to Support Early Design-Stage Sketching of UML Diagrams. In *Proc. of Symposium on Human Centric Computing Languages and Environments, Auckland, New Zealand, October 28-31*, pages 219–226, 2003.

[8] A. Collins and J. S. Brown. Learning issues for intelligent tutoring systems. chapter The computer as a tool for learning through reflection, pages 1–18. Springer-Verlag, New York, NY, USA, 1988.

[9] C. H. Damm, K. M. Hansen, and M. Thomsen. Tool Support for Cooperative Object-Oriented Design: Gesture Based Modeling on an Electronic Whiteboard. In *Proc. of Conference on Human factors in computing systems, The Hague, Netherlands, April 1-6*, pages 518–525. ACM, 2000.

[10] K. M. DeNeve and M. J. Heppner. Role play simulations: The assessment of an active learning technique and comparisons with traditional lectures. *Innovative Higher Education*, 21:231–246, 1997.

[11] S. Georgantaki and S. Retalis. Using Educational Tools for Teaching Object Oriented Design and Programming. *Journal of Information Technology Impact*, 7(2):111–130, 2007.

[12] K. A. Gray, M. Guzdial, and S. Rugaber. Extending CRC cards into a complete design process. In *Proc. of Conference on Innovation and Technology in Computer Science Education, Thessaloniki, Greece, June 30-July 2*, page 226. ACM, 2003.

[13] I. Jacobson. Object-Oriented Development in an Industrial Environment. In *Proc. of Conference on Object-Oriented Programming Systems, Languages and Applications, Orlando, FL, USA, October 4-8*, pages 183–191. ACM, 1987.

[14] R. Lutz, S. Schäfer, and S. Diehl. Using mobile devices for collaborative requirements engineering. In *Proc. of the International Conference on Automated Software Engineering, Essen, Germany, September 3-7*, pages 298–301. ACM, 2012.

[15] N. Mangano and A. van der Hoek. The design and evaluation of a tool to support software designers at the whiteboard. *Automated Software Engineering*, 19:1–41, 2012.

[16] A. Raman and S. Tyszberowicz. The EasyCRC Tool. In *Proc. of International Conference on Software Engineering Advances, Cap Esterel, French Riviera, France, August 25-31*, pages 52–58. IEEE, 2007.

[17] S. Roach and J. C. Vásquez. A Tool to Support the CRC Design Method. In *Proc. of International Conference on Engineering Education, Gainesville, Florida, October 16-21*, pages 1–10, 2004.

[18] D. Sato, H. Corbucci, and M. Bravo. Coding Dojo: An Environment for Learning and Sharing Agile Practices. In *Proc. of the Agile 2008 Conference, Toronto, Canada, August 4-8*, pages 459–464, 2008.

[19] G. Stahl, T. Koschmann, and D. Suthers. *Computer-supported collaborative learning: An historical perspective*, pages 409–426. Cambridge University Press, 2006.

[20] M. F. Wangler and P. Hansen. Visualizing objects: methods for exploring human computer interaction concepts. In *Proc. of Conference on Object-oriented Programming Systems, Languages, and Applications, Vancouver, BC, Canada*, pages 146–153. ACM, 1992.

[21] L. K. Westin and M. Nordström. Teaching OO concepts-a new approach. In *Proc. of the Frontiers in Education Conference, Savannah, GA, USA, October 20-23*, pages F3C – 6–11. IEEE, 2004.