Management, Structures and Tools to Scale up Personal Advising in Large Programming Courses

Jaakko Kurhila and Arto Vihavainen
University of Helsinki
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
Fi-00014 University of Helsinki
{ kurhila, avihavai }@cs.helsinki.fi

Final draft

Originally appeared as: J. Kurhila and A. Vihavainen: Management, structures and tools to scale up personal advising in large programming courses. In SIGITE'11: Proceedings of the 12th annual conference on IT Education. ACM Press, 2011.

Abstract

We see programming in higher education as a craft that benefits from a direct contact, support and feedback from people who already master it. We have used a method called Extreme Apprenticeship (XA) to support our CS1 education. XA is based on a set of values that emphasize actual programming along with current best practices, coupled tightly with continuous feedback between the advisor and the student. As such, XA means one-on-one advising which requires resources. However, we have not used abundant resources even when scaling up the XA model. Our experiments show that even in relatively large courses (n = 192 and 147), intensive personal advising in CS1 does not necessarily lead to more expensive course organization, even though the number of advisor-evaluated student exercises in a course grew from 252 to 17420. A thorough comparison of learning results and organizational costs between our traditional lecture/exercise-based course model and XA-based course model is presented.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education *Computer Science Education*

General Terms Human Factors

Keywords course cost, resource allocation, individual education, conttinuous feedback, instructional design, programming educattion

1 Introduction

We have organized our CS1 courses (and nowadays also other programming and data structure courses) for the last three times using a method called Extreme Apprenticeship (XA). One of the key points of XA is that it emphasizes *doing* over everything else, questioning the utility of lectures, and focuses on active teacher-student collaboration. To be more specific, there are two core values that are stressed in all course activities (adapted from the original description in [16]):

- *The craft can only be mastered by actually practicing it*, as long as it is necessary. In order to be able to practise the craft, the students need to do lots of meaningful exercises. The exercises are designed to build up both *skill* and *knowledge*.
- *Bi-directional continuous feedback makes the learning process meaningful and effective.* It is vastly more efficient if a learner receives even small signals that tell her that she is progressing *and* into the correct direction. In order to give out those signals to the learner, the advisor must be aware of the successes and challenges of the learner. In other words, the advisor must be aware of the student's activities.

The results of applying XA have been impressive in the context of our university, as the drop-out rate, pass rate and grade distribution are all improving[1]. The reasons behind the success stem from the fact that without a prior experience, learning to program has been considered a hard task to master in higher education [15, 3, 14, 12]. Personal XA-based advising sees to it that every individual student practices with tens of simple exercises already during the first week of the course, enforces active participation, and seeks to disable students' ability to procrastinate until the eve of the exam. Learning achievements become visible to the student and internal motivation goes up.

It is easy to believe that organizing XA-style personal learning is straightforward when there is a handful of eager, knowledgeable students and enough competent, hard-working teachers. However, a formal educational organization has its organizational objectives that go beyond a single CS1 teacher; often it means that there are volumes of students with varying backgrounds, i.e. no prior programming experience and relatively low intrinsic motivation toward the subject matter.

This paper describes what kind of human resources and tools are needed in order to enable XA-style education in a large CS1 course. In other words, it is a description of how XA-based education can be scaled up to meet the needs of a formal educational organization that has to serve some two hundred students in CS1. This type of research has been relatively rare and is often only available in a form of policy-level meta-discussion (see e.g. [13]). The empirical evidence behind our attempt is largely based on three separate cycles of XA-based CS1

[1]Data in an unpublished manuscript in preparation [10].

courses[2]. The first cycle of our CS1 courses (Part I and Part II) consisted of 67 and 44 students. When we scaled up the course for Fall 2010, we applied the XA model for significantly larger (192 and 147 students) CS1 courses. In Spring 2011, we applied the XA model with no lectures.

The next section describes the XA method and establishes its value by presenting the improvement in learning results. The remaining sections discuss the issues that arise when one scales up the XA method.

2 Extreme Apprenticeship

Extreme Apprenticeship (XA) builds on Cognitive Apprenticeship [7, 8], a classic model for learning. First, the student is provided with a conceptual model of the process.

Second, students are exposed to tasks (i.e. exercises) that are to be completed under material and advisor *scaffolding*. Scaffolding refers to supporting students in a way that they are not given answers, rather, just enough hints to be able to discover the answers to their questions themselves. Scaffolding works especially well if students are in the zone of proximal development [17]: not too hard, not too easy, just able to do if properly advised.

Scaffolding is faded away when the student starts to master a task.

Practical operation of XA

Many earlier applications on programming education that rely on Cognitive Apprenticeship exist [1, 2, 4, 9]. Extreme Apprenticeship differs from these by the practical issues involved:

1. start with exercises; use small incremental exercises that ensure achievable tasks; exercises need to provide clear guidelines on how to start solving the task and when a task is considered finished
2. exercises define lecture form and content; minimize lecturing and maximize number of exercises
3. advisor must be present in a same space when student is working on the exercises
4. best up-to-date programming practices are emphasized throughout the scaffolding phase
5. students are encouraged to extend their knowledge beyond the instruction provided

Practices 3–5 pose a challenge to the resource consumption and allocation when the number of participants in a course grow. Practice 3 requires added resource consumption, and Practices 4 and 5 require competence from advisors.

[2]In Fall 2010, we used XA-based approach also in our "Computer as a tool" course; in Spring 2011 XA was used partly in our CS2 course (called "Data Structures") and in a new course called "Clojure Programming". The XA-related statistics concerning these courses other than CS1 are out of scope of this paper.

Issues with practices 4 and 5 are out of the scope of this paper, since we did not try to purposefully improve the competence of the advisors, even though it is a benefit for the student if the advisor is competent in versatile ways.

2.1 Extreme Apprenticeship in CS1

Course contents

Our semester-length (6+6 weeks) CS1-type introductory Java programming course consists of two separate parts: Introduction to Programming (part I) and Advanced Programming (part II). Topics covered in both the courses are typical: assignment, expressions, terminal input and output, basic control structures, classes, objects, methods, arrays and strings; advanced object-oriented features such as inheritance, interfaces and polymorphism; the most essential features of Java API, exceptions, file I/O and GUI.

In addition to the exercises, all the study material shown in the lectures is available to students on-line as a web page but written in concise XA style. The material blends both exercises and supporting material, providing students scaffolding as they proceed.

Lectures

As the principles of XA state, lectures are not a necessity in learning to program. This is also evident in our experiments. In Spring 2010, we reduced the number of lectures from the usual 5 hours per week to 2 hours for the first part of CS1, and from 4 to 2 in the second part. In Fall 2010, the responsible lecturer was willing to reduce his lecture hours from 5 to 4 per week. The second part remained the same, 4 hours a week.

It should be noted that there is no minimum in the number lectures in XA-based education. In our Spring 2011 CS1 course, there was only one 1 hour lecture in the whole course (parts I and II combined). Yet the results from the first part are even better than our previous XA-based courses (see the stats in section 2.2).

Exercises

It is expected that students in XA-based courses use most of the time they devote to the course in active solving of programming exercises – either in the computer lab or at home, if the student feels that she has less need for scaffolding. This trains the routine and gives a constant feeling of success by achieving small goals.

For each week a set of new exercises is introduced. We have had number of exercises ranging from 15 to 40. Especially at the start of the course, most of the exercises are small and relatively straightforward. Sequentially completed small exercises combine into larger, more complex programs, which are substantially more challenging than the exercises in our traditional CS1 courses. An added benefit is that combination of smaller parts show the learner how to split a big programming task into sub-tasks.

Exercise sessions with continuous feedback

All exercise sessions need a computer lab or a room with computers with suitable software to conduct the actual programming. XA-based advising is constantly available to the students present during the exercise sessions. Anyone can enter the lab without having to reserve a specific time slot.

If a student does not have specific questions during the exercise session, the advisors are actively observing that the students are working towards the right direction with good working habits, with plenty of verbal constructive feedback.

2.2 Learning results

Comparing the outcomes of the Extreme Apprenticeship -based courses to the previous course instances in terms of percentage of passed students shows clearly that the results have improved after the introduction of XA.

Results are reported separately in the tables below for CS1 part I (Introduction to programming) and Part II (Advanced programming). The XA-based implementations are highlighted in bold face. The column titled n denotes the number of students in each course. The pass-rates are comparable for all the course implementations as the course exams have been kept similar[3]. Significant difference between Spring and Fall semesters that re-occur every year has previously been explained by the fact that courses on Fall semesters consist mostly of CS majors, while Spring semesters consist mostly of CS minors – in XA implementations we have not noticed considerable differences between the skills of minors and majors.

3 Scaling up Extreme Apprenticeship

It is clear that both of the core values of XA (i.e., hands-on practicing and bi-directional feedback) are inherently resource-dependent. Practicing needs a space and a computer with appropriate software; continuous feedback requires advisor input for the student.

The first value proved not to be a problem when scaling up the course. Almost every student of ours has nowadays her own laptop[4]. Most of the computer labs have been dismantled and the remaining few have been heavily underused. Therefore, it was not a problem to just book the labs for the purpose. Moreover,

[3]The low acceptance rate in CS1 part II of Spring 2011 might be explained by the number of tedious experimental exercises that were created by perhaps overly eager TAs – which in turn caused a high drop-out rate. The exam acceptance rate was high: 88 %.

[4]In addition, our department has provided new CS majors a mini-laptop computer at the very beginning of their studies. Albeit not very suitable for programming, they have been used in the XA labs.

CS1 part I

	n	passed
s02	92	38.0 %
f02	332	53.6 %
s03	98	39.8 %
f03	261	64.0 %
s04	84	61.9 %
f04	211	59.2 %
s05	112	46.4 %
f05	146	54.1 %
s06	105	41.9 %
f06	182	65.4 %
s07	84	53.6 %
f07	162	53.0 %
s08	72	58.3 %
f08	164	56.1 %
s09	53	47.7 %
f09	140	64.3 %
s10	**67**	**70.1 %**
f10	**192**	**71.3 %**
s11	**80**	**73.8 %**

CS1 part II

	n	passed
s02	88	26.1 %
f02	249	56.2 %
s03	65	30.8 %
f03	228	59.2 %
s04	66	43.9 %
f04	177	66.1 %
s05	70	57.1 %
f05	125	56.0 %
s06	52	44.2 %
f06	147	67.3 %
s07	53	58.5 %
f07	136	59.6 %
s08	29	51.7 %
f08	147	56.5 %
s09	22	50.0 %
f09	121	60.3 %
s10	**44**	**86.4 %**
f10	**147**	**77.6 %**
s11	**84**	**67.1 %**

Table 1: Learning results. Courses before Spring 2010 have been organized using traditional lecture format.

introductory programming does not require state-of-the-art computers or expensive software. Our lab workstations are equipped with no-cost Linux together with no-cost NetBeans as the pre-installed development environment.

The second value, continuous feedback when practicing, is clearly more difficult to scale up without a direct hit to resource consumption. Moreover, resource usage for continuous feedback is amplified significantly with XA, as there are tens of exercises for every individual. To make this difference more clear, we will present actual numbers from CS1 part I courses: For Fall 2009, our traditional approach typically had 7 exercise groups (of 25 students); every week there were 6 exercises given out. In exercise sessions, one student presented her solution in front of the group and received feedback from the teaching assistant[5]. During the 6-week course, the total number of individual feedback for exercises for all the students combined is 252 (7 * 6 * 6). In the end of the XA-based course in Fall 2010, there were a total of 17420 individually evaluated exercises.

Key solution for overcoming the challenge of resource consumption is to optimize the allocation over time dynamically by using tools, structures, and even voluntary human resources. These issues are discussed in detail next.

[5]In traditional formats at out University, it is also common that when a TA asks for questions, the student do not dare to voice their concerns in a group.

3.1 Dynamic coordination of XA advisors and tools

Dynamic coordination of XA advisors

When we scaled up the XA lab for Fall 2010, we designated one of the advisors as *advisor coordinator*. This was necessary in order to manage day-to-day activities of the XA lab and allocate sufficient resources to appropriate situations. The advisor coordinator had also a final say when recruiting new advisors. The coordinator was a faculty member and received no compensation for the task.

All other advisors worked under the advisor coordinator. The rest of the advisor structure emerged implicitly. All the advisors were compensated equally even though some of the advisors stepped up more than others during the courses. So-called *apprentices*, i.e. fellow students who started to grow into the role of an advisor, emerged during the courses but were not formally recognized nor financially compensated. Some of these apprentices were recruited for the next course as proper advisors, thus enabling continuous flow of advisors to be present.

In traditional courses, teaching assistants (that correspond to advisors in XA-based courses) are compensated with 39 euros/hour. The rationale behind the relatively high pay per hour is that there is a need for preparation for hosting an exercise session. In all XA courses, the advisors are compensated only 17 euros/hr, typically 2-6 hrs per week per advisor. The rationale for the relatively low pay is that the advisors cannot prepare for the XA sessions, as the advisors encounter students' programs fresh in the lab. No advisor complained about the lower per-hour salary, and at no point of time there has been a shortage of very competent students that want to work as advisors.

Many of the advisors are in the early stages of their studies, and their teaching experiences are limited to student tutoring at most. Some were truly novice programmers as they did not have any programming experience outside the CS1 courses. The only common denominator among the advisors is the attitude: ready to work with other students, active and eager to help. Even so, learning results and student feedback has been impressive.

Some of the advisors took a strong role very early in the course but started to fade away towards the end as they felt that their expertise was not sufficient in the latter parts of the course. However, when someone started to fade, there were always advisors who started to step up. This process was "natural" and did not need any management.

Each advisor had the possibility to choose the most preferable time slots for him or her. On-demand service was ensured using IRC[6]. On situations where there were too many students, the advisor were able to ask for extra assistance on-line. We aimed for 1/10 advisor/student ratio in the lab. The communication tool worked also as a fast way to help and share information on problems that a specific advisor himself had faced earlier – similarly the advisors were able

[6] In practise, any instant messaging tool, e.g. Facebook Chat, Google Talk, Skype or MSN Messenger, can be used – our advisors used mainly classic IRC as they were already using it.

to ask for tips on problems they could not help to solve. In addition to IRC, the advisors used text messages and mobile phones to communicate to other advisors. Dynamical resource allocation was welcomed by the advisors.

As *ad hoc* recruitment was practiced, there was no possibility for formal training. Fellow advisors informally and implicitly trained the new advisors. In addition, a "XA lab Manifesto" was established. The manifesto was published in a wiki and updated slightly as experience of proper advising principles grew during the courses. It simply stated few guidelines as pedagogical practices. Note that the practises were worded as personal imperatives, in order to make them more personal:

- You will advise everyone in trouble
- You will not give out solutions but guide the student as much as needed in order to nudge the student to find the solutions herself.
- Advisors do constant round-robin in the lab. Observe and comment on students' progress even if no-one asks anything.
- You will pay attention to the code style: students will learn to program according to Clean Code principles.
- Correct solutions is not enough. You need to push the style towards more understandable and maintainable code.
- Even if there is a slow moment in the lab, you as an advisor cannot sit still minding your own business!

The key issue to keep the lab records correct on a day-to-day basis was an addendum to the XA lab Manifesto, called "Bookkeeping 101". As every advisor understood that XA-based courses can potentially be overly expensive, we communicated clearly the no-waste approach to education and resource usage:

- Prior to course formally agreed lab-hours should be marked down
- If you are alerted to the lab when there is a need for extra advisor, lab-hours are marked down.
- If your lab-time ends but there is less than 10 students remaining and you will stay, lab-hours are marked down. *Note:* Only one advisor will mark down her hours.
- If your lab-time ends but there is less than 5 students remaining, *no advisor* will mark down her hours except in special cases.
- If you are advising in the lab for fun, e.g. when not needed or outside our normal hours, you will not mark down your hours.
- Mark your hours by the end of the day.

Tools

For bookkeeping of student exercises and allocation of advisors, we utilized online spreadsheets in Google Docs with our own macros, which allowed us to keep track of the money spent so far and the demand for advisors during specific times.

In exercise sessions, students had their completed exercises marked down to a check-list, allowing them to see the check-list filled with their completed exercises. We feel that the list played an important role in feedback; every check was an achievement. Check-lists were also updated to the course web-page at the end of every day, allowing students to see the progress of other students as well. In a way, this additional feedback for the students was nearly cost-free, as the records were kept in any case.

4 Results when scaling up XA

Key numbers about the scalability are composed into the tables 2 and 3; *participants* is the number of active students in the course; *eval. exercises* corresponds to the number of exercises that the advisors evaluated from the students; *advisors* is the number of advisors in the advisor pool in that course. The roles and thus their individual working hours varied significantly, from just few hours to 41 hours; *advisor hours* is the total hours the advisors used in total for the course; *total advisor cost* is total cost for advisor salaries during the course; *cost for lectures* is the cost for lectures[7]; *n/a* means that in the initial XA courses advisor hours were not separately tracked. It should also be noted that the advisor coordinator was taking part in the XA labs as tenured faculty developing education, so his salary is not included in XA courses, even though he was actively conducting XA-style advising in the labs.

Fall 2009 courses are traditional and thus based on lectures and exercises; all the other three course instances are full XA-based courses.

term	f09	s10	f10	s11
participants	140	67	192	80
eval. exercises	**252**	6409	**17420**	10648
advisors	5	3	13	11
advisor hours	72	n/a	310	223
total advisor cost	2808	n/a	**5270**	3791
cost for lectures	**3861**	1544	3088	**129**

Table 2: CS1 part I: Introduction to Programming.

We can see from Table 2 that there is a difference in advisor cost (called TA in traditional Fall 2009 course). However, it is more than compensated by the

[7]Lectures in CS1 have been given by tenured teachers, therefore, the cost is hypothetical but based on our actual salary table for the external teachers. In reality, the price tag could be higher as tenured lecturers often do not handle other tasks during courses.

number of exercises (17420) and contact hours (310) the advisors did for the students.

term	f09	s10	f10	s11
participants	121	44	147	84
eval. exercises	**252**	5056	**7349**	9961
advisors	4	7	13	11
advisor hours	72	n/a	271	166
total advisor cost	2808	n/a	**4607**	2822
cost for lectures	3861	1544	3088	0

Table 3: CS1 part II: Advanced Programming.

In addition, Spring 2011 course shows that there is a possibility to save on lectures; in CS1 part I, lectures were cut to only 1 hr (cost of 129 compared to traditional course cost 3088), and in part II, there were zero lectures. Even with minimal lecturing, the course results (Table 1) are higher than ever, even among constantly high-achieving XA-based courses. As stated before and in the principles of XA, lectures are not a necessity.

Because of the increased number of students in Fall 2010, we pre-booked 20 lab hours every week in Fall 2010. In Spring 2010, every week had 8 hours of pre-booked lab time available for the students. In our cost structure, additional lab hours did not pose an additional burden to the budget, since lab reservations can be made free-of-charge. For us, hosting a course entirely in a computer lab would be the cheapest option, as lecture halls and other seminar rooms are rented with high hourly rates. However, money saved by room allocation is not considered in the statistics above, as the rent allocation scheme is a speciality for our university.

Involuntary charity work?

Since the advisors knew the limit for resource usage and could follow the time and money spent into advising, it is possible to think that some of them started to mark down fewer-than-real hours they spent in the XA lab. We can examine the progress of weekly time spent in XA-labs for the advisors (Table 4). It is clear from the figures in Table 4 that this was not the case: the total hours by the advisors correspond to the overall course timeline. Therefore, there is no sign that the advisors felt pressure to do "charity" work, i.e. work without the pay.

Advisor feedback

It is expected that the advisors think that XA requires more work than being a traditional teaching assistant. However, the advisors compared their experiences in XA-style advising to traditional models as "much more rewarding" and "not perceived consuming since it feels so meaningful". Rapid, visible progress of the students was considered efficient use of advisor time. In fact, the advisor

	Week					
Term	1	2	3	4	5	6
f10 - CS1 part I	40	51	56	51	50	50
f10 - CS1 part II	31	45	53	28	47	67
s11 - CS1 part I	37	38,5	34,5	41,5	35,5	36
s11 - CS1 part II	28	31	32,5	28,5	23	23

Table 4: Weekly hours used by advisors.

feedback revealed that the experience was so rewarding, that many advisors volunteered (or "chilled out") in the computer lab and advised the students just for fun.

We sent out a web-form to advisors to collect feedback on the experience on a five-point likert scale[8]. We measured the following dimensions: "rewarding for the assistant" (rewarding), "laborious for the assistant" (laborious), "instructive for the student" (instructive) and "timewise efficient" (efficient). In addition to the previous four dimensions, we presented a meta-question "has improved my own knowledge" (improving). We present answers from only the advisors ($n = 9$) that have been assisting in both traditional exercise sessions and XA sessions. The statistics for the five dimensions are shown in table 5.

Question	Traditional	XA
rewarding	3.22	4.44
laborious	2.66	3.11
instructive	2.88	4.55
efficient	2.44	4.66
improving	3.77	4.44

Table 5: Feedback averages using five-point likert scale when comparing traditional exercise session format with XA exercise sessions.

Table 5 clearly displays the advantage of XA exercise sessions over traditional exercise sessions. Note that all the interviewed advisors had been working in both XA style and traditional exercise sessions.

We also gathered anonymous comments from the advisors. The easy going-feeling of XA sessions is reflected in the following advisor comment.

"XA exercise sessions work as a drop-in-model. You can just walk to the lab and start scaffolding. Exercises are small for the advisor as well, which helps guiding lots of students".

Another advisor reflected students' views in a few sentences, commented on the challenges of being constructive in large groups, and pointed out that students still tend to procrastinate during the weeks.

"Traditional exercise sessions cause far more stress for all parties. As a student one spends energy due to the anxiety of possibly having to go to the

[8] 1: strongly disagree, 3: neither agree nor disagree, 5: strongly agree

front to present your solution, and to understanding the lacks and extras in the presentations from others. As a TA you have an insane judgement- and quality control-role, that cannot be handled easily in a constructive manner for the whole group. In the XA labs it causes frustration that many students want to mark down their exercises during the last days of the week."

5 Conclusions

Extreme Apprenticeship provides a solid structure to organize education that aims to build good routine in programming along with good programming habits such as principles of Clean Code [11] and integrated testing [5, 6]. A key component is that there are advisors who already master part of the craft and are willing to interact with students to help them to grow into expertise. Emphasizing scaffolding in combination with the core values and the derived practices has lead to clearly improved learning results. In fact, the results in learning and the overall feeling towards programming as a tool have helped us at the department to start to re-structure a significant part of our BSc degree courses in CS to benefit from programming. It does not mean that there is a lack of more abstract or theoretical concepts; on the contrary, we have started to see that learning the abstract can benefit from hands-on programming if the student is allowed to "code and play the abstract", not just "see and hear about the abstract". Programming is a helpful tool for most of the issues in the CS education.

It is obvious that hands-on programming practice with timely and constructively helpful feedback needs resources and flexibility in arrangements. Our experiments have shown that even if it is heavy work for all the involved parties (students, teachers and administration), it is possible to receive significant benefits without using significantly more resources. We have been able to match the resources well by adding awareness and interaction between advisors using appropriate tools along with solid processes of organization. We can sum up two principles that we applied when managing XA-based advisor structure in our context — in other words, "Extreme Management":

1. *Known and visible upper boundary for resource usage.* It is vital that every person involved in resource consumption knows the absolute limit for resource use and can view the resource consumption in (semi-)real time. This alleviates the problem that there is the last part of the course going on but all the resources are already used.

2. *Maximum flexibility in organizational structures.* When there are thousands of exercises to be checked by a dozen advisors in one course, not every detail of every aspect of the course is critical. Many of the rough edges (e.g. advisor differences) balance out during the course as there is ample interaction between the advisors and students. In practice, advisors will support each other and do not need to be under explicit supervision.

6 Acknowledgments

Feedback and comments from all the advisors of all the XA-based courses have been invaluable. We want to thank all the advisors (in no particular order) for making XA possible; Aurora, Pekka, Juhana, Antti, Joel, Samuli, Niko, Matti's, Jenny, Jarkko, David, Lasse, Kimmo, Sebastian, Tuomas, Martin, Janne, Joonas, Aleksi – and Thomas!

References

[1] O. Astrachan and D. Reed. AAA and CS 1: the applied apprenticeship approach to CS 1. In *SIGCSE '95: Proc. 26th SIGCSE technical symposium on Computer science education*, pages 1–5. ACM, 1995.

[2] T. R. Black. Helping novice programming students succeed. *J. Comput. Small Coll.*, 22(2):109–114, 2006.

[3] R. E. Bruhn and P. J. Burton. An approach to teaching java using computers. *SIGCSE Bull.*, 35(4):94–99, 2003.

[4] M. E. Caspersen and J. Bennedsen. Instructional design of a programming course: a learning theoretic approach. In *ICER '07: Proc. third international workshop on Computing education research*, pages 111–122. ACM, 2007.

[5] H. B. Christensen. Systematic testing should not be a topic in the computer science curriculum! In *Proc. 8th annual conference on Innovation and technology in computer science education*, ITiCSE '03, pages 7–10, New York, NY, USA, 2003. ACM.

[6] H. B. Christensen. *Experiences with a Focus on Testing in Teaching*, pages 147–165. Springer-Verlag, Berlin, Heidelberg, 2008.

[7] A. Collins, J. Brown, and S. Newman. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In *Knowing, Learning and Instruction: Essays in honor of Robert Glaser*. Hillside, 1989.

[8] A. Collins, J. S. Brown, and A. Holum. Cognitive apprenticeship: making thinking visible. *American Educator*, 6:38–46, 1991.

[9] M. Kölling and D. J. Barnes. Enhancing apprentice-based learning of java. In *SIGCSE '04: Proc. 35th SIGCSE technical symposium on Computer science education*, pages 286–290. ACM, 2004.

[10] J. Kurhila. Carry-on effect in extreme apprenticeship. In preparation.

[11] R. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

[12] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A survey of literature on the teaching of introductory programming. In *ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 204–223. ACM, 2007.

[13] L. B. Resnick and M. Williams Hall. Learning organization for sustainable education reform. *J. American Academy of Arts and Sciences*, 127(4):89–118, 1998.

[14] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13:137–172, 2003.

[15] H. Roumani. Design guidelines for the lab component of objects-first cs1. In *SIGCSE '02: Proc. 33rd SIGCSE technical symposium on Computer science education*, pages 222–226. ACM, 2002.

[16] A. Vihavainen, M. Paksula, and M. Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *SIGCSE '11: Proc. 42nd SIGCSE technical symposium on Computer science education*, 2011.

[17] L. S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes.* Harvard University Press, Cambridge, MA, 1978.