Early Start in Software Coaching

Thomas Vikberg Arto Vihavainen Matti Luukkainen
Jaakko Kurhila

Department of Computer Science, University of Helsinki, Finland,
{tvikberg, avihavai, mluukkai, kurhila} @ cs.helsinki.fi,
WWW home page: `http://www.cs.helsinki.fi/rage`

Final draft
Originally appeared as: Vikberg, T., Vihavainen, A., Luukkainen, M., and Kurhila, J. (2013). Early start in software coaching. In Baumeister, H. and Weber, B. (ed.), *Agile Processes in Software Engineering and Extreme Programming*, volume 149 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer Berlin Heidelberg.

Abstract

The demand for software coaching and coaches is increasing. As our programming courses are organized according to the Extreme Apprenticeship method, it is relatively safe and straightforward to allow students to participate as coaches in our CS1 course even as early as their second semester. Safety is ensured by the hierarchical structure of CS1 course personnel that provides enough peer and faculty support for students undertaking the task of coaching. We briefly describe the Extreme Apprenticeship method as well as the organization and the learning objectives in our coaching environment. Results acquired from student coaches (N=46) indicate that the learning experience of coaching is highly valued and deemed especially educational for the coaches without harming the learning results of the coachees.

1 Introduction

Emergence of lean and agile methods [1, 2, 3] has led to an increasing demand for software engineers that are able to perform as *coaches* for individual developers and teams. To satisfy this demand, higher education institutions with software engineering (SE) education have to give their students opportunities to learn agile coaching skills as well as traditional hard SE skills. As coaching is about working with people [4], learning to coach requires educational structures which involve interaction and cooperation, i.e. opportunities to practice coaching. Time and experience are needed to become an effective agile coach [4]. Therefore, it is beneficial to start practicing it as early as possible, given that supporting conditions can be put in place.

An agile coach not only performs as a teacher, facilitator, collaborator and a mentor, but in addition, an important part is *being* a coach [5]. Coaches guide people on their path towards better expertise through emphasizing best software engineering practices. Acting as a coach requires skills outside the traditional CS degree that consists of e.g. mathematics, programming, databases and architecture design. Agile coaches perform as agents of change and rely upon teamwork-related skills as well as other social skills. These skills are typically embedded only within the "hidden curriculum" within CS degrees which means that their realization is often not assessed or developed.

A traditional approach for coping with the emerging need for coaches in formal higher education would be to offer lecture-based courses titled along the lines of "software engineering coaching". Such courses or modules would introduce the students to e.g. project management methods such as Scrum [1] and Kanban [6] by covering their main principles and practises. Another approach would be to place the students to coach e.g. capstone projects, possibly under the instruction of faculty members of the institution. Such courses can be completed only in the later part of a CS degree as students taking the course should have hands-on experience in larger software engineering projects before they can be put to coach and share the responsibility of capstone-projects (see e.g. [7, 8]).

The approach to software coaching presented in this paper is a mix of hands-on experience with clearly stated learning objectives with a twist: coaching starts very early in the degree programme. The approach is a formal part of the degree. However, it is *not* a course in the traditional sense: there is no lecturing, and no summative assessment of the students; the only course structure is the *specific way that we organize our programming courses.* The course design includes heavy interaction between every participant and a hierarchy of people which allows team-teaching and participation of junior coaches.

Noteworthy is that in our approach to teaching programming, a significant part of our students (ca. 20%) act as coaches in introductory programming courses (CS1) at a very early stage of their studies, as early as their second semester. Working as a coach to novice programmers gives the student valuable experience on technical as well as inter- and intra-personal aspects in programming, e.g. communicating with people with different CS knowledge levels, experiencing and truly understanding the meaningfulness of best programming practices [9]. The coaches are being exposed to thousands and thousands of lines of code from different programmers and are taken into the *community of practice* [10, 11] of a team of coaches. Students are given a chance and an explicit responsibility to see what it can be like to be on the other side of the "teaching podium"; this is expected to both empower the students as well as give insight into what it takes to facilitate learning.

Our approach for organizing coaching opportunities for the students intertwines with our apprenticeship-based method of organizing our programming courses. In the rest of the paper we first briefly describe the apprenticeship-based educational method. We continue by describing the coaching study path offered within our degree, where students are being coached and act as coaches.

We then concentrate on the early coaching experiences of the students, and how the coaching opportunity is organized within our educational setting. Results are gathered and presented from 46 students, who have participated as junior coaches at an early stage of their studies.

2 Extreme Apprenticeship Education

There is a long tradition of apprenticeship-based education in CS, especially in learning to program (see e.g. [12, 13]). As an ongoing effort we have developed a version of apprenticeship-based education called the Extreme Apprenticeship method (XA) that is in use in programming courses at our institution [14, 15]. XA is based on contemporary interpretations of apprenticeship education in which the emphasis is on teaching crafts that require abstract thinking [16, 17, 18].

As is typical for apprenticeship education, XA is based on modeling, scaffolding and fading. First, the student is provided with a conceptual model of the programming process in the form of course material, screen casts and few a lectures. Second, students are exposed to tasks, i.e. exercises, that are to be completed under scaffolding. Here, scaffolding refers to temporary support given to students, which allows them to reach the intended learning objectives. A significant part of the process of scaffolding is given by instructors who perform as coaches for the students. Scaffolding is also built into the learning material and exercises which guide the students to discover the content knowledge that is part of the learning objectives of the programming course.

Students complete programming tasks from day one. They are allowed to experience feelings of satisfaction from completing the programming tasks by themselves. Those giving the support must restrain themselves from giving full answers to the exercises, rather, just enough hints so that the students are able to discover the answers themselves. In XA, the aim is to get everyone to succeed in getting started, and receive enough support to progress further in the course. Many students are spending numerous hours practicing in our XA computer labs. Scaffolding needs to be temporary, and the support given by instructors fades away after it has served its purpose. In the exercise material, this fading means progression to ever larger and more open-ended assignments.

In addition to the adaptations of the three phases described above, XA relies on two key principles: (1) The craft can only be mastered by actually practicing it, as long as it is necessary; and, (2) bi-directional continuous feedback between the learner and the instructor is of utmost importance, in order to make progress and show the progress to both parties [19].

A sufficient amount of practice is ensured by the fact that there are literally hundreds of exercises to be completed. The instructors play a crucial role when interacting with the students. They do not only help the students, but also gather necessary information which is used for continuously assessing the progress of learning in the course. XA relies on this information when the tasks for the upcoming weeks are selected and crafted.

XA has been successfully employed in several courses [19, 20, 21], recently also outside the XA's "home university" [22]. XA has also been adapted to teaching university mathematics [23].

3 Coaching as part of the degree

The software engineering (SE) track at the Department of Computer Science at the University of Helsinki offers various courses that incorporate people skills: SE; SE Project; Software Processes and Quality; Software Project Management and Group Dynamics. Coaching and Engaging in Global Agile Software Teams is a specific coaching course targeted for students at the end of their Master's studies. Topics of the course are agile software development as a concept, agile methodology in distributed settings, and coaching of agile teams taught through problem- and case-based learning techniques. Students also have the opportunity to work under agile coaches in the Master's degree capstone project Software Factory [24]. Due to the technical prerequisites, all of these courses are offered quite late in the studies, see Fig. 1 for placement of the SE courses in the degree programme.

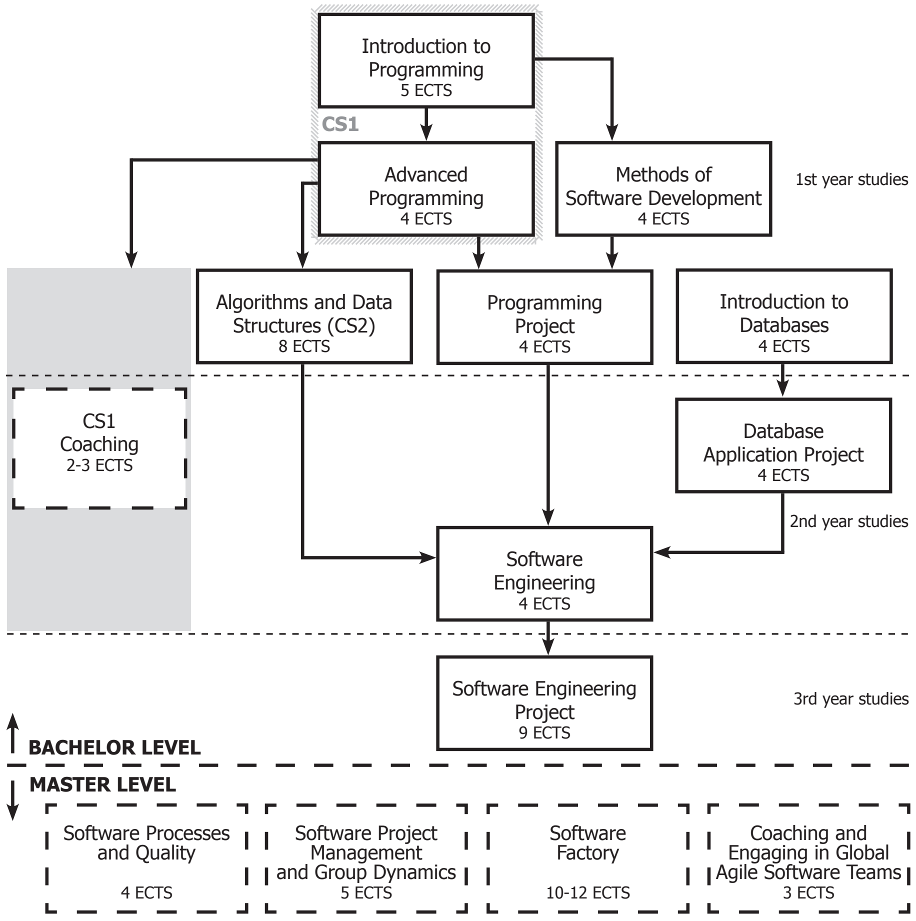
A purpose of the SE subtrack of the department is to educate experts on the path towards *software craftsmanship* [25, 26, 27]. This involves getting the students to focus on software quality, receive a broad understanding of the field and pursue continuous improvement. The courses forming our CS1 are purposefully designed and marketed to be the first steps on a road towards true SE expertise[1]. The emphasis is not only on learning to program, but also on how to program according to industry best practices with the intention to write understandable, easily maintainable and correctly working code. As even the first programming course emphasizes best practices in the industry, it is a suitable setting for our early start in software coaching[2].

In this way coaching is embedded in the curriculum early on. First, students are themselves coached when they participate in CS1. Then they can act as junior coaches in CS1. At this stage, coaching means facilitating the learning of a single CS1 participant.

A broader view of coaching is experienced in the SE Project in which students are coached by the faculty and graduate TAs. At the start of the SE project, the coach of each project group acts in the roles of product owner and scrum master, but during the project they gradually help project participants to take the responsibilities of these roles [20]. After participating in the SE project, many of the students showing interest and capability for coaching are hired as TAs to coach future SE projects.

[1]The CS1 courses can be found as MOOCs at `http://mooc.fi` entitled Object-Oriented Programming with Java [21].

[2]In apprenticeship education, the term "coaching" has been used to refer to the activities of the course teacher (see e.g. [28, 29]). We want to emphasize that coaching in our context is considered a different act from teaching, even if the coaches in CS1 perform as mentors and teaching assistants (TAs) [30]. Coaching is something that the *students* do, in order to learn coaching (in addition to helping fellow students to learn programming).



Introduction to Programming 5 ECTS
CS1
Advanced Programming 4 ECTS
Methods of Software Development 4 ECTS
1st year studies
Algorithms and Data Structures (CS2) 8 ECTS
Programming Project 4 ECTS
Introduction to Databases 4 ECTS
CS1 Coaching 2-3 ECTS
Database Application Project 4 ECTS
2nd year studies
Software Engineering 4 ECTS
Software Engineering Project 9 ECTS
3rd year studies
BACHELOR LEVEL
MASTER LEVEL
Software Processes and Quality 4 ECTS
Software Project Management and Group Dynamics 5 ECTS
Software Factory 10-12 ECTS
Coaching and Engaging in Global Agile Software Teams 3 ECTS

Figure 1: SE-specific courses at the University of Helsinki. The courses with dashed borders are elective courses relevant for coaching. The arrows indicate prerequisites. The figure differs slightly from that in [20] due to minor curriculum updates.

After starting their Master's degree studies, many of the students specializing in SE take part in the Software Factory where they are again coached, this time by experienced agile coaches from our faculty. Besides experiences of coaching and being coached, the Master's degree studies contain many courses on topics related to coaching that give students more opportunities for self-reflection and deepening their theoretical knowledge on the topic. Of course not all CS students follow through the entire coaching track, but those who specialize in SE have the opportunity to experience coaching from multiple perspectives.

This interplay between observing a coach while *being coached* and *acting* *as coach* and thus modeling coaching to others throughout the studies, is fully embedded in the curriculum. It facilitates the progress to become an agile coach as “becoming an agile coach entails education, experience, and practice [...] ’being’ an agile coach in all you do sets a powerful example for everyone you coach” [5].

4 Coaching “course”

The early coaching approach is structured through a format that emphasizes active student engagement over everything else, selection of motivated coaching candidates, and scaffolding of coaches by senior faculty members. The structure is technically a course with study credit to formalize it as a genuine part of a degree. (See Fig. 1 for the position of the course CS1 Coaching within the SE track.) It is important that coaching is a formal activity of the department, as it sets the message that the students are allowed and encouraged to participate, and that they are expected to learn from the experience, even if they are not taught in a traditional sense.

When XA-based CS1 courses began in 2010, the coaching course was not a part of the teaching organization of the course. The students received coaching by teaching assistants selected by the routine selection process of the department. The initial idea of giving students the opportunity to act as coaches as early as possible came from the students themselves. An eager student who had just finished XA-based CS1 approached the faculty in charge of the course and asked – even demanded – to be allowed to help in coaching the students in the next semester CS1 course. After realizing the additional benefits for our evolving CS curriculum SE track, in which agile software engineering principles and practices play a major role [20], the faculty welcomed the voluntary coaches. From the very beginning it was decided that the students’ coaching involvement in the CS1 course was to be organized formally, i.e. students would earn study credits depending on the amount of their involvement.

Initially, this “coaching course” was not marketed nor included in the official study plan. Therefore, the first iterations had only a few participating junior coaches, based on word-of-mouth recruiting. After the initial experiment, the number of students has steadily increased: during the fall 2012 semester we had 26 students as junior coaches and 6 students as senior coaches, coaching our 185 new CS1 students. We aim to have roughly a 1:5 coach-to-student ratio in order to make sure that there are enough CS1 students to be coached in the XA computer labs.

4.1 Connecting coaching to XA

There are clear similarities and synergies between XA education and students as future agile coaches. Fraser et al. state the most important purpose of agile coaching as “facilitating learning” [31]. This same goal is shared by instructors providing scaffolding for students in XA. A good agile coach tries to make herself unnecessary as soon as possible, i.e. helps the team and the team members to flourish [5] bridging directly to the idea of the scaffolding and fading phases in the teaching framework of apprenticeship-based teaching [17, 29].

As XA is a form of apprenticeship education, the "pyramid" of the stakeholders is essential in organizing the CS1 programming course: (1) there are *responsible teachers* (tenured teachers also working as coaches) that are on the top of the pyramid, crafting material and exercises, coordinating and controlling the operation; (2) *senior coaches* (teaching assistants on a payroll) who work as coaches and contribute to exercises in addition to helping the students; (3) *junior coaches* (students taking the CS1 Coaching course), who learn to assist novice programmers by helping students in the XA labs and by being a part of the teaching team; and finally, (4) *students* of the CS1 course (potential junior coaches of future courses).

XA emphasizes individual efforts of students with continuous interaction between all parties, so using XA as a training ground for aspiring coaches is only natural. Junior coaches are typically students in a very early stage of their studies (the CS1 Coaching course can be taken after only one semester of studies); advancing to the stage of a senior coach might take as little as two semesters.

Apprenticeship-based learning stresses the importance of a situative view of learning, which emphasizes that learning activities should take place in the same context as they are practiced [16, 11, 18]. This is also considered in the coaching activities, which take learning of coaching into a genuine environment. All of the stakeholders in coaching, i.e. teachers in charge, senior and junior coaches, form a *community of practice* [10, 11] of coaches for the duration of the course [30]. The community negotiates its meaning by supporting students and actually seeing the results of their participation in coaching.

4.2 Embedding Coaching Course into CS1

The ultimate goal of our SE curriculum is to educate proficient experts in the field of SE. Following the practice of constructive alignment [32] the CS1 Coaching course has its own formal learning objectives[3] that are available for the faculty and students alike. The learning objectives, arranged in a matrix (see Tab. 1), state the principal themes, prerequisites and the evaluation criteria as learning objectives.

The objectives are presented in the form of (1) *approaches the learning objective*, which states the minimum requirements for the activity, (2) *reaches the learning objectives* which mark the requirement of full completion of the course and (3) *deepens the learning objective* which states possible additional objectives and future directions that might be taken into consideration during the activity but are not required.

[3]All mandatory and most of the steadily recurring courses have publicly available learning objective matrices, so students are familiar with them and expect them for every new course.

Table 1: Learning objectives for CS1 Coaching

Principal theme	Enhancing programming skills of peer-students	Instruction skills	Technical tools
Prerequisite knowledge	Good performance in CS1 and capability to produce quality code		Is capable of using VCS and other necessary tools
Approaches the learning objectives	Understands programming code that others have written Notices mistakes in the readability of code written by others Notices mistakes in the design of programs made by others	Is capable of instructing different kinds of people Gives and receives oral feedback Attends scheduled meetings and performs the instruction duties	Deepens the skills to use VCS and other tools Solves CS1 tasks and recognizes different kinds of mistakes in them
Reaches the learning objectives	Recognizes correct solutions of others, even if they differ from own solutions Can instruct mentored students, so that they are capable of correcting their problems with their programming code	Is encouraging Understands that people differ as learners Does not obtrude own solutions, but functions in a learner centered fashion Speaks less than the students Can function as a member of a team of instructors	Recognizes good and bad automated tests
Deepens the learning objectives	Is capable of creating useful tasks and automated tests for CS1 course material	Recognizes factors which help improvement as a teacher Makes students enthusiastic of programming	

4.3 Selection of Coaches

The main requirements for participation in the CS1 Coaching course are the student's willingness to act as an instructor and a decent-to-good grade from CS1, ability to produce quality code and some technical skills (see Tab. 1).

Applications to register as a junior coach are sent well before the start of the term using an online application form. When applying, one needs to type in an open application as well as basic background information such as related grades. If the applicant passes initial filtering (good progress of studies so far), and not known to the course staff beforehand, the applicant is invited for an interview. The interview serves as a guarantee of the necessary people skills needed to be allowed to instruct novice programmers. So far, all applicants have been accepted. As CS1 is organized three times each academic year, it has been possible to allocate most of the junior coach applicants; if not right away, then in the next cycle. It must be stressed that the selected junior coaches become a part of a teaching team and are not expected to perform coaching alone in the XA lab.

Senior coaches are recruited among the more experienced students from the department who have good grades and pace of studies and can show skills in coaching either through performing well as a junior coach, performing well as a teaching assistant in other duties at the department or through work- or hobby-related experience. Unlike the junior coaches, the senior coaches are employees of the department and are therefore subject to standard employement regulations.

4.4 What the Coaches Do

The junior coaches have enrolled voluntarily in an elective SE course. They are motivated and have shown willingness to improve as coaches of novice programmers. This *deliberate practice* [33, 34] of coaching skills is the main tool used to pursue the learning objective of the CS1 Coaching course.

The main task of the coaches is to be a vital part of the scaffolding that makes XA-based programming education possible [29, 19, 30]. The coaches support the students to learn programming by providing individual and interactive feedback to the students. This means that coaches help novice programmers make working software, review their code and point them towards necessary information. An important aspect is that the coaches are expected to embrace the ideas of the students and not obtrude their own solutions on the students, but function in a learner-centered fashion (see Tab. 1). In addition to the soft- and hard-skill related benefits, e.g. communication, experiencing the meaningfulness of best programming practices, we engage our new students in our department's community: the presence of young junior coaches is expected to make the transition from secondary school to the university easier for freshmen students [35].

In addition to the actual coaching, the coaches are encouraged to complete all the CS1 programming exercises before they are released to students. This strengthens their programming routine as well as helps them to direct more time to actual scaffolding instead of wasting time trying to remember what the exercises were about. To increase the formation of the community of coaches, the coaches are encouraged to discuss the exercises with each other, for example in IRC chat. This medium also serves as the main support mechanism for the coaches outside the XA computer lab and the weekly meetings.

An important aspect of the programming exercises of the CS1 course is that the solutions are automatically assessed by an assessment server through a plug-in in the IDE the students use (see [36] for details). The server runs automated tests in order to check for the correctness of the solutions and performs the bookkeeping of the course. This ensures that the coaches can concentrate on coaching, not on trivial correctness checks and error-prone human bookkeeping of student progress.

The junior coaches reflect on the upcoming material and act as beta-testers for the material and exercises by searching for weaknesses in automated tests and inconsistencies in course material. This gives the faculty an opportunity to do improvements before the exercises and course material are released to the course participants. It gives the coaches an opportunity to learn to recognize good and bad automated tests and also leads to a high-quality material as coaches help to make sure that there are no mistakes left in the exercises.

Before the first contact with the students, a 2-hour meeting is organized for the coaches. Faculty members together with the coaches (both junior and senior) go through necessary administrative issues as well as the most important pedagogical practices of coaching CS1. The most crucial information has been gathered in a coaches' set of guidelines and responsibilities that both the coaches and faculty members sign personally.

The guidelines are formulated in an instructive and inclusive manner so that the coaches should observe and proactively coach everyone in trouble. Feedback to the students needs to be constructive and positive. Coaches are instructed to be active even if the students do not ask any questions.

Coaches concentrate not only to the correctness but also to the style of the code: indentation; naming of variables, methods and classes; and method length. In addition, coaches push the students to refactor their code towards a clearer and more maintainable solution so that "the person sitting in the next seat should also understand the solution".

While interviewing the applicants for junior coaches, we have noticed that most, if not all, have a good sense of how an instructor is supposed to scaffold students in the CS1 course. Most of the applicants can already determine what type of coaching is beneficial to students and what is not, e.g. one should only nudge the student towards a correct solution and never give the solution. This is not surprising since all the students applying for the coaching course have so far participated in the XA-based CS1 course themselves.

Despite good prerequisites, the learning objectives could not be obtained without proper scaffolding of the juniors themselves. This scaffolding is conducted by the faculty member(s) in the form of meetings and peer-support. The faculty members as teachers in charge of the course are naturally also present in the XA labs. Students as coaches also perform implicit self-reflection with other coaches while instructing, as well as participate actively in discussions in an online chat.

During the course the responsible teachers organize biweekly face-to-face meetings where all participants of the coaching community are present. Meetings are typically organized as retrospectives, further introducing agile software development practices [3]. In the meetings, the teachers responsible for the course inspect and reflect on what has been done during the past two weeks, and bring up good and bad experiences and practices to the awareness of the whole team. The team identifies top good practices and marks them as *to-be-kept*, i.e. they should work also during the next weeks. Top bad practices are marked as *to-be-improved* that deserve special focus during the next few weeks. The goal is to have a few of the to-be-improved turned into good practices for the future.

5 Results and evaluation

The results show that facilitating software coaching as an early part of a CS curriculum is possible. We started XA-based education in 2010 and the first junior coaches entered the stage in spring 2011. From the fall 2010 to fall 2012 semesters, we have had 101 persons working in our XA labs in different roles. Out of the 101 persons, 78 have served as junior coaches, and 59 as senior coaches (as some have served as both). During the two years, we have been able to offer 5468 hours of targeted hands-on guidance to our CS1 students. This has been done while improving the pass rate [37] and raising the demand level [20] of CS1.

In order to gain some sense of the learning from the junior coaching experiences, we posted a questionnaire to all the students that have been junior coaches in XA-based CS1. We received 46 replies which results in a response rate of 59%. Figure 2 shows the questions and the distribution of the coaches' answers.

One of the main themes of coaching is how to improve the programming skills of students in CS1. This puts some demand on the skills of the coaches themselves. To give appropriate feedback to CS1 students, the coach needs to learn to communicate not only verbally but through written code. This is not only a valuable exercise for the coaches but also intended to make the coaches realize the importance of best programming practices. In the end, if the code is unreadable to others, the maintainability and the functionality of the code diminishes. The learning objective is therefore not only to recognize these problems but to be able to help other students overcome them. We can see in Fig. 2 that "I have become more proficient in reading program code written by others" scores uniformly high (μ=3.96), with low variance (σ^2=.53).

The other main theme of the course is improving coaching and instruction skills. Here, the learning objective is to encounter different kinds of learners and recognize their ways of thinking. This skill should make the junior coaches understand that their own ways of looking at things, e.g. tackling a programming

I have become more proficient ...

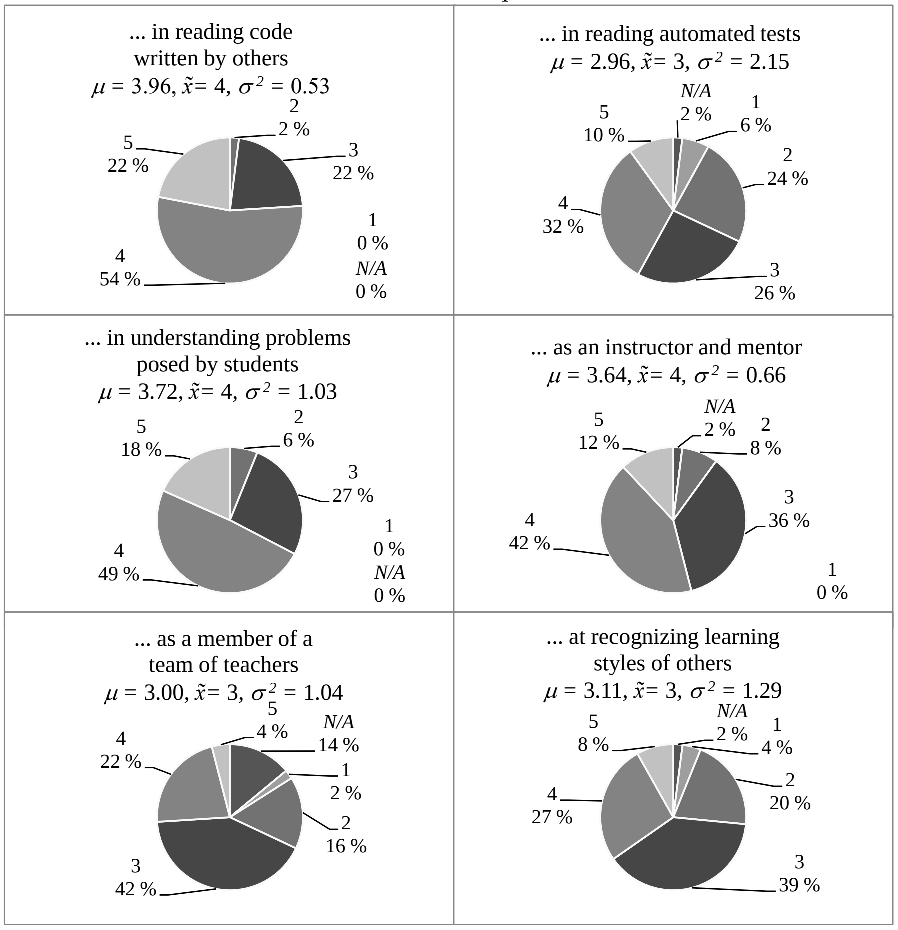


Figure 2: Post-course survey of junior coaches (N=46) with Likert scale (1=not at all, 5=a lot). *N/A* stands for "cannot say" and is not considered in calculating the mean μ and variance σ^2. Median is denoted by $\widetilde{x}$.

task, might differ from those of others. As modern software development is done in teams, an understanding of different working habits and styles is crucial. This is further emphasized by the fact that the apprentices perform team teaching, i.e. coach together along with others, and have to agree with faculty about the practices of coaching and instruction. As we can see in Fig. 2, all of these interpersonal skills score relatively high ("member of a team", "understanding questions", "as instructor and mentor", "recognizing learning styles").

The junior coaches should also deepen their knowledge about the use of professional software development tools. The course material and exercises are maintained using a version control system, and automated testing plays a key role in how the course is formed [21, 36]. We can see in Fig. 2 that "in reading automated tests" the score variance is very high (σ^2=2.15). This is probably due to the varying roles of the junior coaches as some take a more active role in creating and debugging tests. We do not view this as a problem *per se* as it is only natural for coaches to serve in different roles. Automated tests have been used in CS1 material only for a relatively short period of time and are evolving rapidly. It is expected that the role of automated tests will grow in future.

In addition to the Likert scale questions, coaches were able to tell if they had any coaching-related background. As expected, eager coaches tend to have *some* earlier experiences. Even though more than half reported something, background experiences were mostly minuscule, such as a week as a substitute teacher, or group leader in the boy scouts.

Open comments revealed that the coaching experience was highly valued. Quotes such as "coaching is hard but awesome!" crystallize the feelings of many coaches. Other, more targeted open comments included "learning to cope with chaos" and "learning to be more patient", as well as references to "learning to switch fast between mental tasks". Maybe the most insightful comment noted enhanced metacognitive skills:

> Coaching in the XA lab enhanced my view of programming. Afterwards I have noticed how beneficial it was to go through the whole problem solving process from the beginning with another person. That is, from reading the assignment given, all the way to completing the tests. Other people might start to tackle the problem from a completely other angle. This, of course, opens up new paths in my own way of programming.

6 Conclusions

We have been able to give our students an opportunity to act as coaches in a realistic setting at an early stage of their studies. The most important benefit for the students that are participating in coaching very early is the experience that can be used for reflection on upcoming coaching-related courses, as well as when being coached. The experience is also expected to help students to understand the importance of coaching as it pertains to software maintainability, development performance and sensed meaningfulness of software engineering.

In our earlier research, we have seen that the results of CS1 have been significantly improving when using the XA-based approach [19]. Deploying "rookie" students as coaches has not deteriorated the course results. In fact, the number of coaches has allowed us to significantly increase the amount of support and lab times available for our students.

Having a course where students are able to act as coaches during an early part of their studies is only one step. In our curriculum, the students are first coached in CS1, then become coaches, and later on are coached again. This interplay of roles is available throughout the curriculum, and becomes more “real” as the students proceed in their studies. After being a coach for individuals, students are coached as a part of a team, and later on have the opportunity to coach a team.

As coaching is about guiding individuals and teams towards better working practises, it is important that all stakeholders are involved. In our approach, course instructors participate in the coaching activities, and coach the coaches as well as the students. All participants need to spend time in the changing environment to understand the need for change; adaptation should only happen after inspection and only if new practises can bring genuine additional value.

Coaching can be realized by e.g. leading by example, or by gently nudging the participants into a direction, where they are able to realize their mistakes and thus improve. In essense, it is about giving as much freedom as possible while providing scaffolding when needed. The goal is that the coached individuals and teams become self-directed entities that are able to respond to change, and strive to reach their full potential.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. We also wish to thank Mr. Matti Tahvanainen, who was the first eager student to demand the start of this activity as a formal way to be a part of XA-based education.

This work has been partly funded by a grant from the Centennial Foundation of Technology Industries in Finland.

References

[1] Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice Hall (2002)

[2] Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional (2003)

[3] Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change (2nd Edition). The XP Series. Addison-Wesley Professional (2004)

[4] Davies, R., Sedley, L.: Agile Coaching. Pragmatic Bookshelf Series. Pragmatic Bookshelf (2009)

[5] Adkins, L.: Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Addison-Wesley Professional (2010)

[6] Anderson, D.J.: Kanban. Blue Hole (2010)

[7] Hedin, G., Bendix, L., Magnusson, B.: Coaching coaches. In: Proc. of the 4th International Conference on Extreme Programming and Agile Processes in Software Engineering. XP'03, Springer-Verlag (2003) 154–160

[8] Hedin, G., Bendix, L., Magnusson, B.: Teaching extreme programming to large groups of students. Journal of Systems and Software **74**(2) (2005) 133–146

[9] Martin, R.C.: Clean Code: A Handbook of Agile Software Craftsmanship. Robert C. Martin series. Prentice Hall (2009)

[10] Wenger, E.: Communities of Practice: Learning, Meaning, and Identity. Learning in Doing Series. Cambridge University Press (1998)

[11] Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. Learning in Doing. Cambridge University Press (1991)

[12] Astrachan, O., Reed, D.: AAA and CS 1: The applied apprenticeship approach to CS 1. SIGCSE Bulletin **27** (1995) 1–5

[13] Kölling, M., Barnes, D.J.: Enhancing apprentice-based learning of Java. In: Proc. of the 35th SIGCSE Technical Symposium on Computer Science Education. SIGCSE '04, New York, NY, USA, ACM (2004) 286–290

[14] Vihavainen, A., Paksula, M., Luukkainen, M.: Extreme apprenticeship method in teaching programming for beginners. In: Proc. of the 42nd ACM Technical Symposium on Computer Science Education. SIGCSE '11, ACM (2011) 93–98

[15] Vihavainen, A., Paksula, M., Luukkainen, M., Kurhila, J.: Extreme apprenticeship method: key practices and upward scalability. In: Proc. of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education. ITiCSE '11, ACM (2011) 273–277

[16] Brown, J., Collins, A., Duguid, P.: Situated cognition culture of learning. Educational Researcher **18**(1) (1989) 32

[17] Collins, A., Brown, J., Holum, A.: Cognitive apprenticeship: Making thinking visible. American Educator **15**(3) (1991) 6–46

[18] Collins, A., Greeno, J.G.: Situative view of learning. In Aukrust, V.G., ed.: Learning and Cognition. Elsevier Science (2010) 64–68

[19] Kurhila, J., Vihavainen, A.: Management, structures and tools to scale up personal advising in large programming courses. In: Proc. of the 2011 Conference on Information Technology Education. SIGITE '11, ACM (2011) 3–8

[20] Luukkainen, M., Vihavainen, A., Vikberg, T.: Three years of design-based research to reform a software engineering curriculum. In: Proc. of the 13th Annual Conference on Information Technology Education. SIGITE '12, ACM (2012) 209–214

[21] Vihavainen, A., Luukkainen, M., Kurhila, J.: Multi-faceted support for MOOC in programming. In: Proc. of the 13th Annual Conference on Information Technology Education. SIGITE '12, ACM (2012) 171–176

[22] Dodero, G., Di Cerbo, F.: Extreme apprenticeship goes blended: An experience. In: 12th IEEE International Conference on Advanced Learning Technologies. (2012) 324–326

[23] Hautala, T., Romu, T., Rämö, J., Vikberg, T.: Extreme apprenticeship method in teaching university-level mathematics. In: Proc. of the 12th International Congress on Mathematical Education, International Commission on Mathematical Instruction (2012)

[24] Abrahamsson, P., Kettunen, P., Fagerholm, F.: The set-up of a valuable software engineering research infrastructure of the 2010s. In: Workshop on Valuable Software Products. (2010)

[25] McBreen, P.: Software Craftsmanship: The New Imperative. Addison-Wesley Professional (2001)

[26] Martin, R.C.: The Clean Coder: A Code of Conduct for Professional Programmers. Robert C. Martin Series. Prentice Hall (2011)

[27] Luukkainen, M., Vihavainen, A., Vikberg, T.: A software craftsman's approach to data structures. In: Proc. of the 43rd ACM Technical Symposium on Computer Science Education. SIGCSE '12, ACM (2012) 439–444

[28] Bareiss, R., Radley, M.: Coaching via cognitive apprenticeship. In: Proc. of the 41st ACM Technical Symposium on Computer Science Education. SIGCSE '10, ACM (2010) 162–166

[29] Caspersen, M.E., Bennedsen, J.: Instructional design of a programming course: a learning theoretic approach. In: Proc. of the 3rd International Workshop on Computing Education Research. ICER '07, ACM (2007) 111–122

[30] Vihavainen, A., Vikberg, T., Luukkainen, M., Kurhila, J.: Massive increase in eager TAs: Experiences from extreme apprenticeship-based CS1. To appear in: Proc. of the 18th Annual Joint Conference on Innovation and Technology in Computer Science Education (July 2013)

[31] Fraser, S., Lundh, E., Davies, R., Eckstein, J., Larsen, D., Vilkki, K.: Perspectives on agile coaching. In: Agile Processes in Software Engineering and Extreme Programming. Volume 31 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 271–276

[32] Biggs, J., Tang, C.: Teaching for quality learning at university: what the student does. Society for Research into Highter Education. McGraw-Hill (2007)

[33] Ericsson, K.A., Krampe, R.T., Tesch-Romer, C.: The role of deliberate practice in the acquisition of expert performance. Psychological Review **100**(3) (1993) 363–406

[34] Litzinger, T.A., Lattuca, L.R., Hadgraft, R.G., Newstetter, W.C.: Engineering education and the development of expertise. Journal of Engineering Education **100**(1) (2011) 123–150

[35] Clark, M., Lovric, M.: Suggestion for a theoretical model for secondary-tertiary transition in mathematics. Mathematics Education Research Journal **20** (2008) 25–37

[36] Vihavainen, A., Vikberg, T., Luukkainen, M., Pärtel, M.: Scaffolding students' learning using Test My Code. To appear in: Proc. of the 17th Annual Joint Conference on Innovation and Technology in Computer Science Education (July 2013)

[37] Vihavainen, A., Luukkainen, M.: Results from a three-year transition to the extreme apprenticeship method. To appear in: Proc. of the 13th IEEE International Conference on Advanced Learning Technologies (July 2013)