

Bringing Test-Driven Software Development to University Classrooms

Morten Goodwin Christian Auby
Per Arne Andersen Vera Barstad

{morten.goodwin,christian.auby}@uia.no
{vera.barstad,perara12}@gmail.com

Faculty of Engineering and Science, University of Agder
Serviceboks 509, NO-4898 Grimstad, Norway

Abstract

The software development industry has gone through major changes these last years with more focus on agile and test driver development methodologies. This has made the daily life of programmers vastly different from how students typically learn programming at university levels. To accommodate for this discrepancy, this paper introduces an end-to-end method and tool-set for teaching software development with agile and test-driven focus. The students will in their learning environment work with the same type of tools and methods used by the industry, making them familiar with test driven development during their studies.

In the proposed solution, the teacher works as a master by providing tests for each assignment the students need to carry out. Subsequently, each student commits their code to a git repository, builds their solution at a central server and runs the corresponding tests. They get automated responses on whether they have carried out their assignment correctly or not.

The solution is tested out successfully by students and teacher, and is on the verge of being deployed in a real teaching environment.

1 Introduction

Teaching programming classes is virtually unchanged since the introduction of software engineering at university level. Typically, the teacher lectures on a subject and the students carry out related assignments. Whenever the teacher has time s/he evaluates the assignments and provide feedback. In contrast to the uniformity of university lecturing, the industry has dramatically changed their development methodologies to better fit customers high demands and rapidly changing requirements. Most programming students will, when they start in the industry, work with agile and test-driven development methods [1]. There are several problems with the discrepancy between university teaching and the

This paper was presented at the NIK-2014 conference; see <http://www.nik.no/>.

methods used in the industry. First and foremost, the programming classes are out of sync with the industry we are trying to prepare the students for. Secondly, all students are required to follow the teaching pace, which typically yields a uniform pace of the class — in a way striving for everyone to be average. Thirdly, the students will only get feedback on their assignments when the teacher has time to evaluate them. Thus, they are reliant on the teacher making time available for evaluation.

The agile development process does not directly fit into traditional university pedagogics. In a university classroom there are typically one teacher (and multiple teacher assistants), and hundreds of students. The agile development cycle relies upon, among others, close collaboration between those doing the programming — the students — and those that give the task — the teacher. This is, to put it mildly, impractical with large university classes due to the limited human resources and time. For example, a meager five minutes of one-to-one interaction between a teacher each student in a class of 180 students sums up to 15 hours of time weekly.

This paper introduces a tool-set to bring the agile software development world into large university classrooms. By utilizing standard industry solutions, this paper presents an end-to-end mechanism for submitting assignments and automatically evaluating these in a test-driven environment. This makes the students operate in an agile process, preparing them for what awaits in the industry, as well as getting immediate feedback on their work without reliance on teacher resources.

Outline

The remainder of the paper is outlined as following. Section 2 presents an overview of state-of-the-art on automated student assignments and bringing agile and test-driven development into university classrooms. Section 3 introduces an end-to-end mechanism using agile industry tools for development. Section 4 continues with discussion on the findings based on the presented approach and tests with actual students. The paper ends with conclusion and future work in section 5 and 6.

2 State-of-the-art

Learning and mastering practical technical subjects, such as software development and programming, is challenging for many students. How to best teach programming is addressed many times in the literature. Among the most effective means of teaching programming is one-to-one teacher/student interactions [2, 3, 4], and hands-on feedback [5]. This enables immediate feedback on any programming task carried out by students, including information on to what extent the program is well written or whether there are improvement potentials in the code.

Automated student interaction

To be a good programmer, as in many other fields, students need to train in large volumes [6]. However, due to the number of students participating in programming courses, the “best practice” teaching approach is difficult to execute in actual university classrooms. There are simply not enough time available for close interactions with every student. Mitigations for this problem are available in the literature. Among others, several interactive student tools for programming courses are proposed [7, 8]. One of the well known examples is the tool “Independent Student Tutoring by Examining Program” [9]. This tool was used as part of an introductory programming course, and the authors did

several double studies and reached the indisputable conclusion that students with the tool need less help from the teacher and were able to finish their task quicker than the control group. In fact, introducing the tool reduced the time the students spent on finishing their task down to less than one third. Hence, they showed indisputable evidence that their interactive tool helped students in learning programming.

Even though interactive teaching tools for programming are available in the literature, they have profound limitations [3]. Most often the tools are kept within the university where it is made and not made available to the public. In addition, the tools are typically university centered, meaning that it developed specifically to interact with university learning management systems (LMS). This means that the students become familiar with a specialised system completely different from what they will experience in their working career [10]. There is, to the best of our knowledge, little emphasis on making automated student interaction tools connected with existing industrial standards.

Agile tools in teaching

Many universities teach development methodologies including the unified process and more agile developments such as XP and Scrum. There are even universities that embrace the agile methodologies as a pedagogical methods [11, 12, 13, 14].

One representative study on agile teaching of programming was as following [11]: The teacher acts as a manager — similar to a scrum master — and provides the students with a document explaining the module, a set of templates, and a zero release project. During the project, the teacher plays an active role and evaluates each student deliverable, both collectively and individually. There are typically many deliverables per student every week which takes up the teachers time. In fact, the authors note that 85% of the teacher's time is spent on laboratory work. They further note that the total time the teacher spend on a class was reduced by 20% by introducing their tool. The reason for the reduction is caused by the teacher pro-actively spending time on addressing issues up front and in turn reducing the amount of time needed to be spent answering student questions. However, it is notably so that the amount of time spent on a class when you have one-to-one interaction must be directly related to the size of the class. In the current study, the course was carried out with only 25 groups. Hence, it is arguably so that the time spent by the teacher will increase in even larger student groups.

Others have carried out case studies using agile software development for extracurricular activities [15]. Among their main conclusion was that for students with little or no programming background, which is to be expected in a lower level programming class, too much teacher resources is spent on guiding the students.

Additionally, pair programming, which is a key aspect of the agile methods, is applied in several studies in order to increase the programming experience [16]. This work was motivated by the lack of constructive alignment theory in programming courses. They found that such pair programming has a significant flaw in a teaching environment namely that it enables the students to bypass the pedagogical intent of the course.

Several researchers have also examined agile learning environments in order to present an agile pedagogy. This includes exploring to what extent agile learning environment enhance the learning in programming courses and if it improves the critical thinking skills of students [17].

MOOCs

The use of Massive Open Online Courses (MOOCs) is a trend among large and small universities. Perhaps what gets most attention is the large well respected universities — such as Harvard, Stanford and MIT — successfully applying MOOCs in addition to their classical education models [18]. The courses available through MOOCs include everything from physiology to economics and introductory programming courses. Some work exists on automated and semi-automated assessment for exercises including software development [18, 19], but much focus is on peer assessment. In peer assessment environments, the students are required not only to deliver assignments, but also review and provide feedback on other students hand-ins.

Most research focus in MOOCs is on technical and pedagogical challenges [20]. To the best of our knowledge, there is little or no work on making MOOCs industry relevant including merging it with test-driven development methods.

Agile development tools

Many tools are used in agile development, of which the most relevant for this paper is tools version control and continues integration.

Version control

Proper use of version control accelerates and simplifies the software development process [21]. There is no doubt that *git* is the current go-to version control system, far superior to SVN and CSV on speed and merging possibilities.

Stash is a *git* server that allows an administrator to create and manage project and source code repositories [8]. The main administration page is a web interface, where each project can have any number of repositories, and access can be managed on a per project or per repository basis.

Build tools for continuous integration

Continues integration is that developers continuously build the software throughout the development cycle. By using continues integration, developers will detect problems early in the development process and thus over all increase the quality of the code. Several build tools for continues integration exists, such as Maven, Jenkins and Bamboo [22].

Bamboo is a continuous integration server whose primary purpose is to build projects as changes come in, and optionally run any unit tests that are defined in the project.

3 Approach

This section presents Test Centered Learning Tool (TCLT), an end-to-end method and tool-set to provide a test-driven environment to students in university programming classes.

Figure 1a presents the internal architecture of the system.¹ For each course using the tool-set, there are two main projects created: a Stash project and a Bamboo project [24]. Stash and Bamboo are chosen because they are well established industry leading software architecture tools [25, 26], they seamlessly fit the projects need for industry relevant version control and build options, and they natively support each other.

¹Part of the system already existed in a previous internal project called Fjelltools [23]. This project was carried out in collaboration between industry and university partners.

In TCLT, users interact directly with both Stash and Bamboo. This section explains how Stash is used to serve *git* repositories and combine the repositories of tests (accessed by teacher) and student hand-ins (accessed by students) in meaningful ways. The section continues with explaining how Bamboo is correspondingly used for the students and teacher to build and test the application. This section rounds off with architecture specific details.

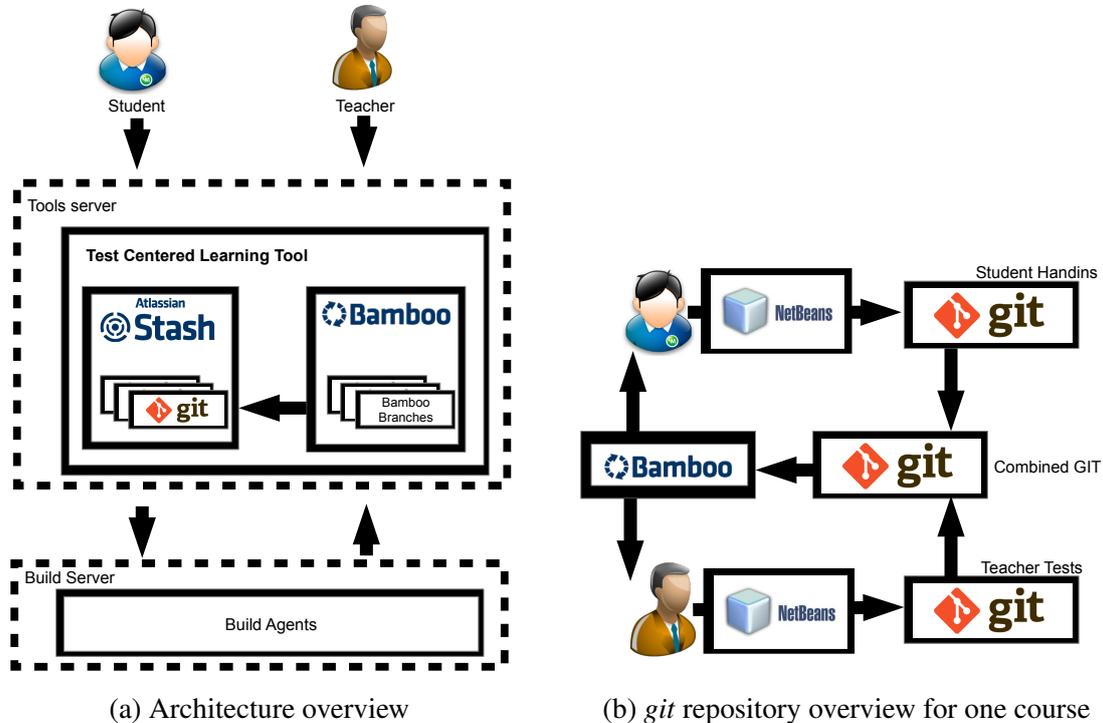


Figure 1: Architectures

Git repositories through Stash

For each course three main *git* repositories are created. An overview can be seen in Figure 1b:

1. **Teacher-git:** *git* repository where the course responsible commits unit tests for each assignment.
2. **Student-git:** *git* repository where the student commits assignment hand-ins.
3. **Combined git:** *git* repository that combines the two above repositories to apply the unit tests.

Prior to any student work, the teacher makes tests for each assignment available in the Teacher-*git*. When the students carry out the assignments, they commit and push their hand-ins to their Student-*git*, which is the formal deliverables. During the build process (see Figure 1b), the student- and teacher-*git* is merged in into the combined *git* one branch per student. This enables testing of the student assignments without student access to the unit tests.

An important concept is that a successful student hand-in will build successfully and pass all unit tests given by the teacher, while an unsuccessful hand-in will have one of

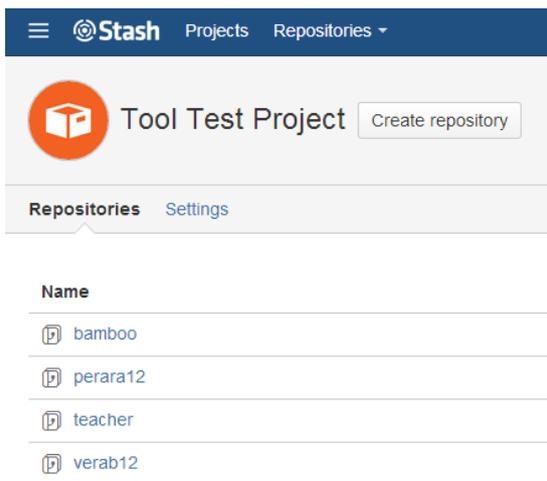
more failing tests. Hence, the students can know whether or not they have successfully carried out an assignment by simply running the unit tests. The proposed methods allows for this to be possible with limited additional work from both the teacher and student.

Figure 2a presents a screen shot from the Stash tool as it is seen when a teacher logs in. In this example, there are four git repositories: The teacher git (teacher), the combined git (bamboo) and two student git repositories (perara12, verab12).

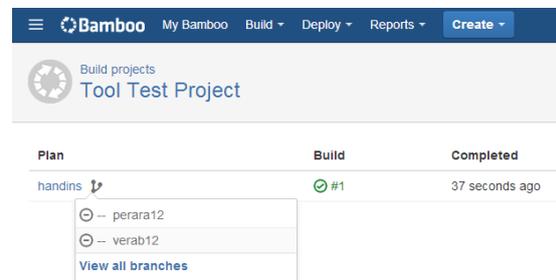
Test secrecy

Each student has his/hers own git repository per course with all course hand-ins for that course available. This repository is considered private and only the student and the teacher has read/write access to it. The students do not have read/write access to neither each other’s repositories, the Teacher-git nor the Combined git.

The reason for this strict policy is that it is always possible to “cheat” unit tests . Even though the Teacher-git does not include the solutions, only the tests, in certain situations it can be tempting for the students to find the simplest way that will make the unit test pass without doing the actual assignment. This can often be done by closely examining the unit tests, and in this way by-passing the actual assignment. To avoid such problems, the course responsible have the option of keeping not giving the students read access to the Teacher-git. Naturally, if the course responsible desires, the Teacher-repository with the unit tests can be made available to the students.



(a) Stash



(b) Bamboo

Figure 2: Screen shots

Building process

The building process is basically compiling and testing the committed code. The procedure is a following. The course responsible, acts as a master, and develops a bamboo build plan, student assignments and corresponding unit tests. The build plan resembles a Unix Makefile and include information of which git repositories to use, how to compile the files, and so on.

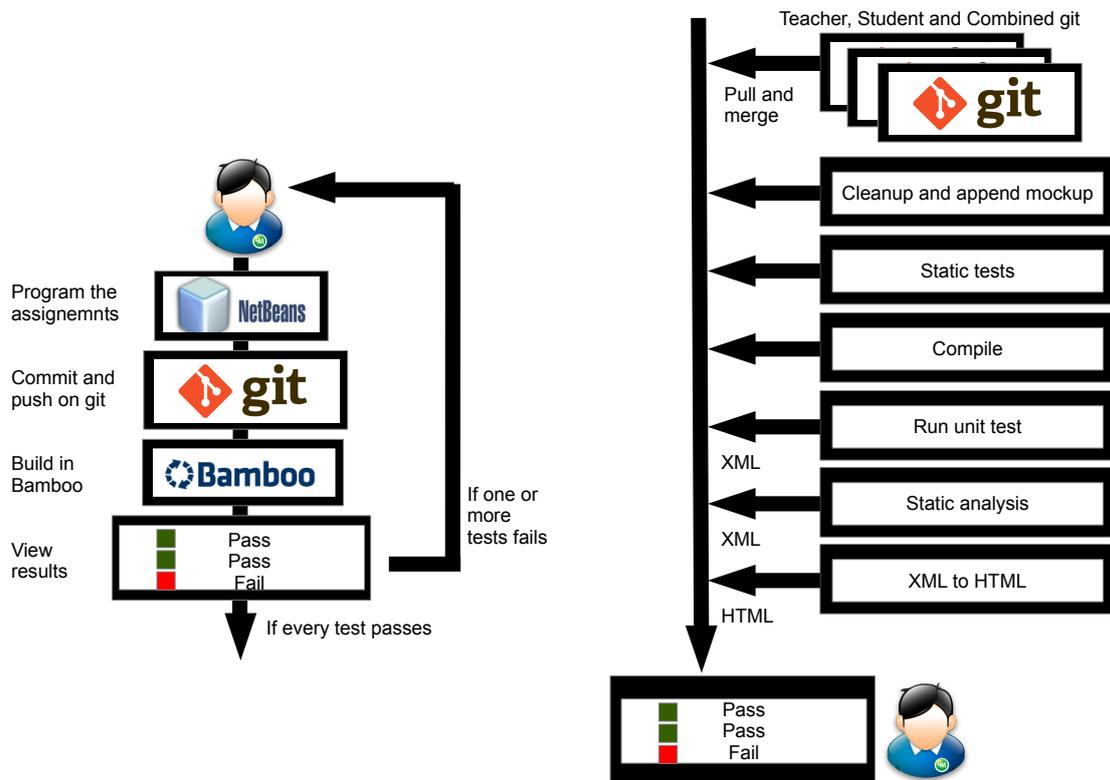
After both the teacher have committed their tests and the students have carried out the assignments, the students can build and test their hand-in through the build server Bamboo.

Bamboo build plan

The Bamboo project consists of a plan that builds all sources code (student hand-ins and tests), and runs the tests. By default one plan builds one branch on one repository, but this becomes tedious if you have many different branches that needs to be built automatically. Bamboo has a feature called plan branches, which, when enabled, automatically creates a new plan for each branch based on a parent plan. TCLT makes use of this feature by making the course responsible develop one plan per course, and creating one empty branch per participating student. This giving each student their own plan branch, which are copies of the main course plan. A script in the course uses the branch plan variable to check out the correct repository for each student.

When a student wants to build and evaluate their hand-in, s/he enters his/hers branch, and data from the Teacher-git and Student-git are checked out into a new combined git, as illustrated in Figure 1b. In the combined git, each student will have a separate git branch which is automatically created based on the student user name. Subsequently, the build process compiles the needed files, install needed dependencies, and run unit tests on the file. An XML-file is then created following a standard XMLUnit format. Finally, he XML-files is parsed and presented as readable HTML for the students.

An example illustration of build plan for a basic Java programming course can be seen in Figure 3b, and a screen shot of Bamboo with two Bamboo branches, perara12 and verab12, is seen in figure 2.



(a) Student process for handing in and building assignment

(b) Example of Bamboo build plan

Figure 3: Build plan

Student workflow

Figure 3a shows the build process from the student point of view. This is as following:

1. Do the required assignment.
2. Commit and push the files.
3. Build the hand-in (including tests from the teacher) on Bamboo (press play in the Bamboo interface).
4. Read the results.
5. If all tests pass, you are finished with the assignment, otherwise you need to go to the first step.

This organisation gives several benefits, among others that the students get instant feedback on how they are doing. Further, students are not able to read the teacher git and can therefore not cheat by knowing the details of the unit tests.

Architecture

Bamboo needs at least one agent that knows how to run the plan defined in the Bamboo project. A dedicated server has been set up to run virtual build agents (see *build server* in Figure 1a). Each hand-in is evaluated on one build agent, prevents student tests from slowing down the *tools server* by for example running infinite loops. It further makes it possible to restrict network access on the *build server* as a whole.

4 Results and Discussion

This section presents results and discussion on the TCLT method and tool-set. This section starts with presenting the anticipated learning outcome, benefits and challenges using this method, and continues with a test case for verifying TCLT. Finally, this section analyses the results from the test case.

Learning outcome

The proposed methods, TCLT, is meant to be an end-to-end solution for students and teachers in programming courses.

Since the method is designed with test-driven development in mind, it answers to the need for system development students to learn about testing [27]. Because the students work in a test-driven environment, they will implicitly be familiar with both the general test-driven methods, as well as experience with practical tools such as version control and build tools.

Further, due to the fact that the methods relies upon automation, student hand-ins are automatically evaluated. From this, additional benefits emerges. The students are able to continually interact with TCLT, as shown in Figure 3a. The students can carry out an assignment, commit and push the code to git and build the application. If the tests pass, they know that their implementation is correct and can continue with the next assignment. On the other hand, if one or more tests fail, they are informed that they have done something wrong and need to redo the assignment. This continues until all their tests pass. Hence, the students will automatically be guided towards the correct solution.

This also makes it possible for the students to work any hour of the night without reliance upon the teacher. The teacher may give many more assignments with little resource increase. Because the responses are automated, there is no requirement for a manual evaluation of the student hand-ins.

It is important to keep in mind that this automated approach does not eliminate teacher interactions. The automated feedback will only give information on whether the solution is correct or wrong, the students will still require teacher feedback to understand how to correctly implement the solution.

Student tests

For testing the approach with actual students, a basic Java programming course was used as a case. 11 students who had recently completed the basic programming course volunteered to test the solution. The participating students had little or no knowledge of git or Bamboo, nor programming experience beyond their completed course.

The students were allowed to use any IDE of their choice as long as they were able to commit on git. However, detailed instructions, both textually and a video lecture, were given to them that based itself on NetBeans. Thus, most participants choose to use NetBeans.

The students were given two basic Java assignments each consisting of 6 smaller assignments to complete; The first assignment was creating a simple application with dialog boxes asking for a name, and then printing the same name. The second assignment, shown in Figure 4 was more complex on developing an advent calendar using the Java Swing library. The assignments were deliberately designed to be relatively easy for the students, but still complex enough to test the tools set and their experience with it.



Figure 4: Screen shot from student hand-in

Findings and student feedback

Of the 11 students participating, only one was able to complete the task without teacher help. A conclusion to be drawn from this is that the students will need close guidance in the first time they use git and Bamboo.

Further, several of the students ran tests from the wrong Bamboo branch. Instead of running their own branch, they built the master-branch. Unfortunately, this was the branch where the teacher had pushed the solution. A direct consequence was that the students wrongfully thought they had completed the task without problems. In hindsight, the detailed description could easily be misinterpreted to run the wrong branch.

Some students chose to use Eclipse and egit instead of NetBeans. These students had problems with the “fast forwarding” overwriting their solution. From their experience, they finished the first assignment successfully. However, when they wanted to commit and push the second assignment, they overwrote existing code. By first glance, it looked as if the first hand-in was deleted, and they suddenly failed the first hand in. This means that special attention on how the students use git needs to be considered.

Other than the above mentioned situations, the test was a success.

5 Conclusion

This paper presents an end-to-end tool-set, Test Centered Learning Tool, integrating test-driven development into university programming classes. The tool-set utilizes well established frameworks in the industry including git, Bamboo and Stash.

The tool enables students to work in industry like environments already in their education. By following the proposed method, they will implicitly learn essential parts of test-driven development from the very beginning and, as a direct bonus, receive automated feedback on their assignments.

The student case test clearly indicates that several aspects needs to be specially considered. For students who are not familial with version control and build tools, as can be expected at as start of university study, there needs to be careful guidance through the required steps.

6 Future work

The most significantly future work is that the set of tools is planned to be deployed in a real setting. This is scheduled to happen the following semester (fall 2014).

Additional future work includes making student feedback with focus on pedagogical content. In the current implementation, the students only get pass or fail for each assignment. A more pedagogical approach would be to, whenever a student fails a assignment, provide resources such as web pages, videos and chapter references. This gives the students a concrete step forward on what to work with in order to pass the test.

There is also a plan for more advanced features, including a peer assessment approach. This enables not only automated assessments of the student hand-ins, but also reviewing of each others work — which is in line with agile development. In addition, work is planned on learning each students weakness and deploying assignments accordingly. Finally, there is a plan to further embrace pedagogical motivational techniques such as gamification [27, 28]. The students could for example get points per completed assignment, move on to the next level, and so on in order to motivate.

7 Acknowledgements

This paper is part of Test Centered Learning Tool, <http://tclt.uia.no> and is an ongoing internally funded research project at University of Agder.

References

- [1] Tore Dybå and Torgeir Dingsøy. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833 – 859, 2008.
- [2] Elizabeth Odekirk-Hash and Joseph L. Zachary. Automated feedback on programs means students need less help from teachers. *SIGCSE Bull.*, 33(1):55–59, February 2001.
- [3] Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005.
- [4] Dana P Leonard, Jason O Hallstrom, and Murali Sitaraman. Injecting rapid feedback and collaborative reasoning in teaching specifications. In *ACM SIGCSE Bulletin*, volume 41, pages 524–528. ACM, 2009.
- [5] Alexander Hoem Rosbach and Anya Helene Bagge. Classifying and measuring student problems and misconceptions. *Norsk informatikkonferanse (NIK)*, 2013, 2014.
- [6] Anca Deak and Guttorm Sindre. Analyzing the importance of teaching about testing from alumni survey data. *Norsk informatikkonferanse (NIK)*, 2013, 2014.
- [7] Riku Saikkonen, Lauri Malmi, and Ari Korhonen. Fully automatic assessment of programming exercises. In *ACM Sigcse Bulletin*, volume 33, pages 133–136. ACM, 2001.
- [8] Stephanie Rogers, Steven Tang, and John Canny. Acce: automatic coding composition evaluator. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 191–192. ACM, 2014.
- [9] Elizabeth Odekirk-Hash and Joseph L Zachary. Automated feedback on programs means students need less help from teachers. In *ACM SIGCSE Bulletin*, volume 33, pages 55–59. ACM, 2001.
- [10] Stephen H Edwards. Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In *Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA*, volume 3, 2003.
- [11] Maria Isabel Alfonso and Antonio Botia. An iterative and agile process model for teaching software engineering. In *Software Engineering Education & Training, 18th Conference on*, pages 9–16. IEEE, 2005.
- [12] Dennis P Groth and Edward L Robertson. It’s all about process: project-oriented teaching of software engineering. In *Software Engineering Education and Training, 2001. Proceedings. 14th Conference on*, pages 7–17. IEEE, 2001.
- [13] Anthony J Cowling. The crossover project as an introduction to software engineering. In *Software Engineering Education and Training, 2004. Proceedings. 17th Conference on*, pages 12–17. IEEE, 2004.

- [14] Ahmed Seffah and Peter Grogono. Learner-centered software engineering education: From resources to skills and pedagogical patterns. In *Software Engineering Education and Training, 2002.(CSEE&T 2002). Proceedings. 15th Conference on*, pages 14–21. IEEE, 2002.
- [15] Stephen R Foster and Ruben Ruiz. Constructivist pedagogy meets agile development—a case study. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 1092–1096. IEEE, 2010.
- [16] Alessio Gaspar and Sarah Langevin. An experience report on improving constructive alignment in an introduction to programming. *Journal of Computing Sciences in Colleges*, 28(2):132–140, 2012.
- [17] Morgan C Benton and Nicole M Radziwill. A path for exploring the agile organizing framework in technology education. In *Agile Conference (AGILE), 2011*, pages 131–134. IEEE, 2011.
- [18] Steve Cooper and Mehran Sahami. Reflections on stanford’s moocs. *Communications of the ACM*, 56(2):28–30, 2013.
- [19] Gergely Bertalan and Michael Rumler. Moodle: Question types: javaunittest, June 2014.
- [20] Li Yuan, Stephen Powell, and JISC CETIS. Moocs and open education: Implications for higher education. *Cetis White Paper*, 2013.
- [21] Stefan Otte. Version control systems. *Computer Systems and Telematics Institute of Computer Science Freie Universität Berlin, Germany*, 2009.
- [22] Chris Geiger, Dennis Przytarski, and Sascha Thullner. Performance Testing in Continuous Integration Environments. Master’s thesis, Institute of Software Technology, University of StuttgartRijksuniversiteit Groningen, Universitatstrasse 38, 70569 Stuttgart, 2014.
- [23] Christian Auby. Fjelltools. Internally published at University of Agder, 2010.
- [24] Stefan Kohler. *Atlassian Confluence 5 Essentials*. Packt Publishing Ltd, 2013.
- [25] K Sigerud and V Baggiolini. The software improvement process-tools and rules to encourage quality. In *Conf. Proc.*, volume 111010, page THBHMUST04, 2011.
- [26] Robert Varga. Changing dashboard build system to bamboo. Technical report, 2013.
- [27] Swapneel Kalpesh Sheth, Jonathan Schaffer Bell, and Gail E Kaiser. Increasing student engagement in software engineering with gamification. 2012.
- [28] Michael Blumenstein, Steven Green, Ann Nguyen, and Vallipuram Muthukumarasamy. An experimental analysis of game: a generic automated marking environment. In *ACM SIGCSE Bulletin*, volume 36, pages 67–71. ACM, 2004.