Assessing Iterative Practical Software Engineering Courses with Play Money

Kai Mindermann, Jan-Peter Ostberg and Stefan Wagner
University of Stuttgart
Institute of Software Technology
{kai.mindermann|jan-peter.ostberg|stefan.wagner}@informatik.uni-stuttgart.de

ABSTRACT

Changing our practical software engineering course from the previous waterfall model to a more agile and iterative approach created more severe assessment challenges. To cope with them we added an assessment concept based on play money. The concept not only includes weekly expenses to simulate real running costs but also investments, which correspond to assessment results of the submissions. This concept simulates a startup-like working environment and its financing in an university course. Our early evaluation shows that the combination of the iterative approach and the play money investments is motivating for many students. At this point we think that the combined approach has advantages from both the supervising and the students point of view. We planned more evaluations to better understand all its effects.

CCS Concepts

•Social and professional topics → Software engineering education; *Student assessment;* **•Software and its engineering →** *Agile software development; Programming teams;*

Keywords

Practical course, play money, finance, iterative

1. INTRODUCTION

Teaching software engineering includes theoretical and practical parts. The practical parts give students the possibility to apply their recently gained knowledge of software engineering aspects in mostly artificial tasks.

The practical software engineering course is then often the first course that forces the students to **work on a real-world problem in a team**.

In our case, the bachelor degree study path at the University of Stuttgart, the practical course has diverse goals. First it is meant to let the students engineer a new application from the beginning, starting with requirements analysis through design, implementation and test, to a final customer acceptance test. Second it is meant to let them experience and cope with team dynamics and the problems that come inherently with that.

ICSE '16 May 14-22, 2016, Austin, TX, USA

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4205-6/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2889160.2892660

2. GENERAL CONDITIONS

The task in each years practical course is to **engineer an application** defined by a customer which itself is often a (changing) local company. The students work in **teams of 3**. Each **team works independently** on the same task and gets assessed by an assigned tutor individually. The teams have approximately **13 weeks** to complete all required milestones and deliver a final product.

The practical course was previously organized as a waterfall model. For every phase the students had to submit corresponding artifacts. That approach gave them the ability to experience a full life cycle of software development. But the non-flexible waterfall model made it hard to cope with different paces of the developer teams and milestones had to be rescheduled for many teams individually. That also required very flexible tutors and it increased the effort for the supervisors, who had to grant each rescheduling. Also it was difficult to determine and communicate when a team has failed to satisfy the requirements to pass the practical course.

Also, due to the distinct transition of more and more companies and teaching to agile software development methods, we wanted to address that in the practical course as well. A change with comparable results was done by Bruegge et al. with a much more complex environment [1] based on earlier experiences with a comparable course size and goals [2].

3. CHANGES

The changes resulting from the reorganization do not only concern the software development method but also some surrounding conditions as well. We replaced the strictly predefined phases of the waterfall model with an iterative approach and added a play money based realistic assessment model.

3.1 Iterative Development

The previous waterfall model was replaced by a 1-month long initial requirements and software design phase, two 4-week long iterations and one final week for a customer acceptance test. The initial idea was to offer three 1-month long iterations in which the students could do requirements analysis, design and implementation unsupervised. The problem is, as this is the first time for the most students to do a whole project, they lack experience. Previous practical courses have shown that students need guidance in the design phase to be able to successfully develop a fitting application. So we decided to have only 2 iterations and an explicit setup phase before the first iteration.

We suggested that the iterations consist of a short planning phase and then the implementation. But during the iteration the students were free to organize themselves and work as they seem fit. Only the final submission after each iteration had to be fully tested and implemented features with corresponding test coverage reports.

Additionally we defined 7 milestones. The most milestones define fixed dates at which the teams have to submit their current results. The milestone 3 is after the setup phase, milestone 4 after the first iteration and milestone 5 after the second iteration. The milestones 0 and 2 require only attendance and milestone 6, the customer acceptance test, requires both attendance and a final submission. An important thing is that milestones can not be postponed. If a team can not deliver the required artifacts it can only deliver them at one of the following milestones under the conditions of the following play money based assessment.

3.2 Play Money Based Assessment

To make the course more realistic and inspired by real-world startup company financing, each team gets a **virtual play money account** with an initial investment of 15,000 € (5,000 € for each developer). During the course each team has **weekly** *running expenses* based on their teams size. 1,000 € *labor costs* per developer and 1000 € *fixed costs* which together usually make **4,000 € per team per week**. At most milestones the submissions are assessed by a tutor. They decide, based on that assessment, how much play money will be invested in that team. A team can only continue the practical course, if it has a neutral or positive balance after a milestone.

Due to the initial investment almost every team can *survive* up to milestone 3 without doing anything, so the pressure is not as high as the running expenses suggest.

The virtual play money account is represented by wiki page in the used social coding platform GitLab[1]. It shows *detailed consecutive bank statements* for the team.

We have 3 general categories for which the tutors can invest money:

1) Documentation: The documentation consists at least of the product backlog with its user stories and the software design draft.

2) Critical Features: Implemented, tested and working functionality which was specified as critical by the supervisors and the customer.

3) Additional features: Implemented, tested and working functionality which improves or extends upon the critical features in some useful way.

We also have a predefined range for the amount for each investment. We carefully simulated different possibilities for the team performance and the corresponding investment. These simulations let us adjust the range of the investments. For example: We may invest between 0 € for no or insufficient, 4,000 € for bad and up to 20,000 € for outstanding documents. This fine-tuning was needed to distinguish between *good and bad* team performance after the assessment of the last milestone simply through their final play money balance.

For the correct calculation and compliance with the predefined investment amounts we implemented a spreadsheet which is used by the tutors during the assessment. The supervisors keep an overview through an online spreadsheet that contains the closing balance for each team at each milestone. It is updated by the tutors.

[1]https://about.gitlab.com Accessed 2016-01-12

4. EVALUATION AND EXPERIENCES

We expected some teams to delay working on the tasks until the first iteration because of the rather big initial investment. But we observed none. What we observed was that some could not comply with the submission format requirements, which was simply a *pushed git-tag* that marks a specific version of their repository as submission. Those teams got no investment, as was known from the documentation, at milestone 3. But most of those teams did it right the next time and got the investment for their whole work at milestone 4.

After the first iteration but before releasing the assessment of that iteration, we gave every participant of the practical course a detailed questionnaire[2]. 75 of 104 completed it. The questions were mostly matrix-questions with 5 options ranging from *very applicable* to *not applicable* and some free text questions. The most distinct and interesting answers are the following:

Approximately 60 % considered the play money assessment concept understandable and approximately 50 % considered the play money assessment concept realistic and gained additional motivation from it.

The intended separation between the critical features implementation in the first iteration and the implementation of additional features in the second iteration was accepted as reasonable by approximately 70 %. Also approximately 70 % agreed with the selection of the critical features through the supervisors and the customer.

5. CONCLUSIONS

The new process for the practical course not only makes supervision easier and frees up time for tutors and supervisors, which can be invested in deeper running support, but also it is more realistic. Moreover, as the first evaluation shows, it is motivating for the students.

As the practical course is in progress during the writing, we plan a second evaluation at the end to get even more insight into our approach to teaching iterative software development and assessing it.

6. REFERENCES

[1] B. Bruegge, S. Krusche, and L. Alperowitz. Software engineering project courses with industrial clients. *Trans. Comput. Educ.*, 15(4):17:1–17:31, Dec. 2015.

[2] B. Bruegge, H. Stangl, and M. Reiss. An experiment in teaching innovation in software engineering: Video presentation. In *Companion to the 23rd ACM SIGPLAN OOPSLA*, 2008.

[2]http://dx.doi.org/10.5281/zenodo.45967