This is the author prepared accepted version. © 2010 IEEE. Published in:
David Broman. Should Software Engineering Projects be the Backbone or the Tail of Computing Curricula?. In Proceedings of the 23th IEEE Conference on Software Engineering Education and Training (CSEE&T 2010), Pages 153-156, Pittsburgh, USA, 2010. https://ieeexplore.ieee.org/document/5463585

Should Software Engineering Projects be the Backbone or the Tail of Computing Curricula?

David Broman
Department of Computer and Information Science, Linköping University
david.broman@liu.se

Abstract

Most computer science (CS) and software engineering (SE) curricula include some form of SE project with the aim of lowering the gap between CS and SE education and real-world demands in industry. In this paper we briefly discuss and explain our findings of software engineering projects taught at LiU in Sweden. These courses use what we call the "tail" approach, where student projects are performed at the end of a degree program. We then argue that there are several problems with this approach and sketch an idea where SE projects are an integrated part of a curriculum. Furthermore, pros and cons of this idea, which we call the SE project "backbone" approach, are discussed and analyzed.

1. Introduction

Teaching both theory and practice of software engineering (SE) is hard. Not only because of the width and complexity of the subject, but also because of the following inherent catch-22 teaching problem: On the one hand, for students to really be able to understand and reflect on SE theory, they need to have some experience of working in a "real" project. On the other hand, to learn successfully from an SE student project course, it is vital to have a sound understanding of the basic principles and practices from SE theory. Feedback from the software industry have shown an increasing demand for CS students with skills in areas such as testing, review techniques, release management, and team work [3]; all areas that are common topics within SE. The ACM/IEEE-CS Software Engineering curriculum guidelines [4] emphasize that the curriculum should incorporate a significant portion of real-world elements to enable learning of skills and concepts within SE. Two kinds of project types are suggested:

- *Project-based classes*, including elements such as group-work, presentations, formal reviews etc.
- *Capstone course* that is a significant project typically spanning over the last year of a degree program, and aims at practicing the student's skills and knowledge that he/she learnt during the earlier years.

The latter approach of having a larger project at the end of a degree program is both common and often referred to as a successful approach to meet the industry's demands [2,5]. However, is this the most efficient way of training students to be professionals? Should larger software engineering projects only be located at the end of a degree program (at the tail), or are there other possibilities to design a curriculum that may result in more skilled and mature students that are better prepared for an industry career? In this paper we discuss these questions and present them as follows:

- We give a brief overview of our own experience of teaching SE courses, discussing both benefits and problems using this "tail"-approach (Section 2).
- An idea where software engineering projects form the "backbone" of the degree programs is suggested. We then analyze and discuss how this approach addresses some of the problems introduced in the tail-approach (Section 3).

2. Software Engineering Projects as the Tail

Currently, there are two different SE project courses taught at Linköping University (≈200 students). In one of the courses, students in teams of 8 develop software products for a real customer in industry. In the other course, the focus is on processes and less on technically challenging projects. In this course, 30 students are working together in a simulated company consisting of two departments with cross-functional teams. Both courses are in the later years of the programs (3rd or 4th year).

2.1 Experiences from the Current Approach

We have during the last 3 years improved the structure and execution of the SE projects regarding different aspects. The evaluation of the project setup with a simulated company structure is still work in progress, but our findings so far indicate that this way of organizing an SE project is appealing and motivating for students. However, one of the main challenges concerns getting enough resources for supervising the students. Another common challenge is the ability to assess and grade student project work achievements. We believe that it is important to not only judge the outcome of the students work, i.e., to judge artifacts such as documents and source code base. Instead, emphasize should be put on the *process* of their work, i.e., how they work, not only what is produced. In the earlier years project work, student feedback indicated that the students did not understand why they created different artifacts; it was only produced because of the fact that supervisor requested them. Hence, we believe that the SE project education should have a practical driven approach [8], i.e., a "practice-pull" approach where e.g., documents and other artifacts are created because the students can see that they must exist and are useful to them for making a good job in the project.

2.2 General Problems with the Tail Approach

There are still several inherent problems with this way of structuring a curriculum:

- *Breadth of skills*. By running a realistic large project once, each student can try one role, e.g., project manager, lead developer, analyst, or tester. Hence the student learns a lot about the skills for their particular role, but does not get the breadth of skills from different roles. Similar problems with breadth of skills in capstone projects have been reported by Karunasekera and Bedse [5]. A possible workaround could for example be role rotations during the project. However, such an approach may give confusion within the teams and lowers the realism in the project work.
- *Limited resources for supervision*. Teaching SE project courses are in general expensive, since good supervision or coaching require high amount of face-to-face time with the students. In large classes, such as our course with 120 students in the project course, large personnel investments are needed. Extensive contact with students is even more important if grades should be handed out, especially if it should be given on an individual and not team basis.
- *SE experience requires maturity*. To reach a high level of maturity within the field, experience from different projects and different situations is needed, together with time for reflection on related SE theory. With the tail approach, where the large project is performed at the end of the education, it is assumed that the student can increase his/her maturity level in a short time frame, with experience from only one project. We believe that this is insufficient for reaching a high level of maturity and professionalism at the end of a degree program.

3. Software Engineering Projects as the Backbone

It ought to be a general consensus within the SE community that software development is an iterative process, i.e., that it is very hard to "do things right" from the beginning. Hence, it might come as a surprise that the common belief seems to be that it is enough for students to do one large project at the end of a degree program, i.e., that the students can "learn the right things" from the beginning, without prior knowledge from working in a larger project.

3.1 The Basic Idea

Instead of having a capstone project at the end of a degree program, the basic idea is to have a new project running every year during all years in a program. The following list shows potential properties of this approach:

- *Company approach.* We have found that running the project as a larger simulated company (approximately 30 students) gives the students new insights and questions when developing larger scale software. However, to also stimulate a creative working atmosphere, it is still important to form smaller cross-functional teams (7 +/- 2 persons).
- *Constrained budget.* We successfully used this model for many years, where students create time reports and have a limited amount of total time in the project. This gives a general insight of the "real world" where you always are working with a constrained budget.
- *Participants from all years*. Since the project course is running every year, the aim is that a company consists of participants from all years of the program. Interaction between students from different years can give a creative learning environment. Younger students can learn from the older students, and the older students improve their in depth understanding of specific subjects. The best way to learn something in depth is to teach.
- *Real customers from industry.* We believe that the customer in the projects should be real customers with real requirements. However, it does not need to be only one customer; the product can be a generic and suitable for many different customers. Hence, the customer role can be several different companies, each giving input to potential requirements for the product.

3.2 Discussion

The obvious criticism to the above suggested change to a CS or SE curriculum is if this is realistic at all? Would not this kind of course demand large portion of the total student's available time and reduce the amount of time for other subjects? The answer is both yes and no. Yes in the sense that time for traditional teaching with lectures and written examination will undoubtedly be lower. However, that does not mean that the learning objectives in even the theoretical subject necessarily will be lower. The suggested approach promotes active learning where the teacher is not the central figure; the "sage on the stage" [6]. Instead, the teacher should act as a coach that helps and guides the student in the learning process. We believe that the student only can construct new knowledge, based on its current alternative framework [1]. Hence, e.g., theory lectures in SE subjects such as project management, requirements engineering, testing theory, and system design should be available for students from different years, emphasizing the "practice-pull" approach for learning. Examination and courses in other subjects should of course be taught in parallel, but several of the more practical components of courses could be integrated with the company's activities. This might be

challenging to do in practice, but we are convinced that it will give highly motivated students with increased learning capability.

One implication of students coming from different years in the program is that they naturally are assigned different roles depending on their experience and skills. Forth and fifth-year students could preferably take leading management roles such as lead developer, lead analyst, and project manager. First year students should perhaps be limited to simpler tasks such as basic testing or limited programming tasks. However, this might result in the risk that certain roles, e.g., testers get the reputation as less skilled roles. Hence, it is very important that the supervisors take an active part in the project to eliminate such misunderstandings. Moreover, different levels of promotion within the company organization can also generate both incentive and motivation [7].

The most important aspect of the backbone approach is that students from different years in the program participate in the same project. This increases diversity in the company and may also result in that students get a more mature and pragmatic view of problem solving. This is also the single concept that addresses all the problems in Section 2.2. For example, since students take the course several times, and get "promoted" and switch roles between the years, the likelihood that the students get a breadth of skills is higher. Moreover, since more experienced students help the novice ones, the burden on supervision is mitigated, even if it neither can nor should be removed completely. At Linköping University, a curriculum called *innovative programming* has been using courses with integration of students from different years, but not yet to the extent as suggested in this paper.

4. Conclusions

We have in this paper discussed several problems with the so called "tail"-approach, where a project is located in the end of a degree program. An idea was introduced, where the project part forms the backbone of the whole curricula. To conclude, we believe that the backbone approach must be further analyzed and detailed, and it is our hope that this paper can be a starting point for such discussion.

Acknowledgements

The author wants to thank Kristian Sandahl and Peter Fritzson for discussions and comments. This work was funded by Vinnova (OPENPROD project) and VR.

References

[1] M. Ben-Ari, "Constructivism in computer science education", In proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, page 257-261, ACM Press, 1998

[2] A. Goold, "Providing Process for Projects in Capstone Courses", In Proceedings of the 8th annual conference on Innovation and technology in computer science education, Pages 26-29, ACM, NY, USA, 2003

[3] Interim Review Task Force, "Computer Science Curriculum 2008 – An Interim Revision of CS 2001", ACM/IEEE-CS, 2008

[4] Joint Task Force on Computing Curricula, "Software Engineering 2004 – Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering", ACM/IEEE-CS, 2004

[5] S. Karunasekera and K. Bedse, "Preparing Software Engineering Graduates for an Industry Career", In Proceedings of the 20th Conference on Software Engineering Education & Training, IEEE, 2007

[6] A. King, "From Sage on the Stage to Guide on the Side", College Teaching, Vol 31, Issue 1, Heldref Publications, Washington DC, USA, 1993

[7] M. Moore and C. Potts, "Learning by doing: Goals and experiences of two software engineering project courses", Software engineering education. LNCS vol. 750, Springer, 1994

[8] L. Ohlsson and C. Johansson, "A Practical Driven Approach to Software Engineering Education", IEEE Transactions on Education, Vol. 38, No. 3, IEEE, 1995