Three Years of Design-based Research to Reform a Software Engineering Curriculum

Matti Luukkainen, Arto Vihavainen, Thomas Vikberg
University of Helsinki
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
Fi-00014 University of Helsinki
{ mluukkai, avihavai, tvikberg }@cs.helsinki.fi

ABSTRACT

Most of the research-oriented computer science departments provide software engineering education. Providing up-to-date software engineering education can be problematic, as practises used in modern software development companies have been developed in the industry and as such do not often reach teachers in university contexts. The danger, and often the unfortunate reality, is that institutions giving education in software engineering end up teaching the subject using outdated practices with technologies no longer in use. In this article we describe a three-year design-based research where the goal has been to design and reform a software engineering subtrack within our bachelor curriculum that would make it possible for the students to have strong up-to-date theoretical and practical skills in software engineering without a need to remove any of the existing theoretical aspects.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation, Management

Keywords

instructional design, modern software engineering practises, computer science curriculum

1. INTRODUCTION

Bachelor level computer science (CS) education has many goals. A CS curriculum should offer students a solid theoretical foundation in CS as well as convey enough practical skills and knowledge on current technologies. Although the focus of many CS departments in research universities lies in theoretical fields, such as algorithmics and computational complexity, most universities embed software engineering (SE) studies in their curriculum. This poses a great challenge for SE educators as they need to keep up with the rapid development of the IT industry. The danger, and often the unfortunate reality, is that institutions giving SE education end up teaching SE using outdated practices with technologies no longer in use.

According to IEEE Software Engineering is *(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)* [1]. In contrast to more established fields of CS, the field of SE is still rather immature and has seen a radical paradigmatic change in the past decade with the emergence of agile methodologies [2, 7]. In contrast to traditional plan-based methods (e.g. the so-called Waterfall method [6]), agile methodologies emphasize working software, customer collaboration, human interaction and best technological practices.

It has lately been recognized that SE is more a craft than a science [5] and as such it is to be treated differently than the more theory-based topics. The up-to-date industrial best practices and tools should be taken into account when teaching SE. It has been noticed that with a traditional university instruction students even need to unlearn working methods acquired in formal education when entering the industry [3, 4].

One of the major problems is that most university teachers lack recent industrial experience or direct contact to industry, and as such are not aware of the emergent modern SE practises such as agile methods. As teachers have not used them, they are not truly capable of conveying agile methodologies in teaching as agile methodologies can often only be learned through practicing them [7].

These issues have led to a situation where the SE-related bachelor level education is heavily out of synch with reality. Teaching yesterday's theories poses problems for the credibility of the education and leads to a lack of motivation amongst students with existing industrial experience.

The main SE learning objective of a graduating bachelor at our department is that the student should be able to perform efficiently in a modern software development project. In this paper we describe a three-year design-based research where the goal has been to design a SE subtrack within our CS bachelor curriculum that would make it possible to have strong up-to-date theoretical and practical skills in SE.

SIGITE'12, October 11–13, 2012, Calgary, Alberta, Canada.

2. REFERENCES

[1] A. Abram, J. W. More, B. Pierre, and D. Robert, editors. *Guide to the Software Engineering Body of Knownledge*. IEEE Computer Society, 2004.

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

[3] A. Begel and B. Simon. Struggles of new college graduates in their first software development job. In *Proceedings of the SIGCSE '08*. ACM, 2008.

[4] A. Fox and D. Patterson. Crossing the software education chasm. *Communications of ACM*, 55(5):44–49, may 2012.

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

[6] W. Royce. Managing the development of large software systems. In *Proceedings of IEEE WESCON 26*. TeX Users Group, August 1970.

[7] K. Schwaber and M. Beedle. *Agile Software Development with SCRUM*. Prentice Hall, 2002.