Empirical Software Engineering: Teaching Methods and Conducting Studies

Claes Wohlin

Blekinge Institute of Technology, Box 520, SE-37225 Ronneby, Sweden
Claes.Wohlin@bth.se

Abstract. Empirical software engineering has grown in importance in the software engineering research community over the last 20 years. This means that it has become very important to also include empirical studies systematically into the curricula in computer science and software engineering. This chapter presents several aspects and challenges to have in mind when doing this. The chapter presents three different educational levels to have in mind when introducing empirical software engineering into the curricula. An introduction into the curricula also means increased possibilities to run empirical studies in student settings. Some challenges in relation to this is presented and the need to balance educational and research objectives is stressed.

1. Introduction

Empirical software engineering has established itself as a research area within software engineering during the last two decades. 20 years ago Basili et al. [Basili86] published a methodological paper on experimentation in software engineering. Some empirical studies were published at this time, but the number of studies was limited and very few discussed how to perform empirical studies within software engineering. Since then the use of empirical work in software engineering has grown considerably, although much work still remains. A journal devoted to empirical software engineering was launched in 1996 and conferences focusing on the topic have also been started. Today, empirical evaluations and validations are often expected in research papers. However, the progress in research must reflect on education. Computer scientists and software engineers ought to be able to run empirical studies and understand how empirical studies can be used as a tool for evaluation, validation, prediction and so forth within the area.

This chapter presents some views on education into empirical software engineering. In particular, the chapter contributes by identifying three different educational levels for empirical software engineering. Furthermore, the chapter stresses the possibility to run both experiments and case studies as part of courses and how this facilitates research. However, to perform research as part of courses, with the educational goals of a course in mind, requires a delicate balance between different objectives. The research goals must be carefully balanced with the educational goals. Different aspects to have in mind when balancing the different objectives are presented. Finally, the need to develop guidelines for running empirical studies in a student setting is stressed. Guidelines are needed both to ensure a proper balance between different objectives and to, within the given constraints, get the maximum value from empirical studies run in a student setting. It is too simplistic to disregard empirical studies with students just because they are students instead there is a need to increase our understanding of how empirical studies with students should be conducted and to what extent results from them could be generalized.

The chapter is outlined as follows. Section 2 provides an overview of some related work. In Section 3, three different educational levels to consider when introducing empirical studies into the curricula are presented, and different types of empirical studies to use in relation to courses are discussed. Section 4 presents some aspects to have in mind when balancing educational goals and research goals. Finally, a brief summary and conclusions are provided in Section 5.

2. Related work

The literature on education in empirical software engineering is limited. The literature primarily describes either specific experience from a course, such as [Höst02], or from performing a research study with students as subjects [Thelin03]. Most empirical studies conducted in an educational setting are controlled experiments, although some articles are more related to studies of student projects and hence could be classified as case studies [Höst02, Berander04]. Some exceptions exist, where authors discuss the use of students in empirical studies for different purposes [Carver03].

Anyway, there is a need to improve the way we both teach and conduct empirical studies in software engineering. However, there are many challenges. Software engineering is primarily concerned with large scale software development and hence, for example, the use of controlled experiments in laboratory setting is not straightforward. This is true both when it comes to student learning and to conducting research in an academic setting. We have to understand and define how to conduct controlled experiments in laboratories to make them useful in a larger context. This makes the challenges quite different from other disciplines using experimentation. There is a lot to learn from other disciplines, but there is also a need to address the specific challenges when conducting experiments in software engineering.

When experimentation was introduced into software engineering, the main focus was on running experiments. As the work progressed, more focus has been put on the actual methods used. The two books on experimentation [Wohlin99, Juristo01] are good examples. Researchers have also started addressing how results from different empirical studies should be combined either through meta-analysis [Hayes99, Miller99] or using a more evidence-based approach [Kitchenham04]. Other researchers have addressed the challenge of building families of experiments [Basili99] and hence plan for combining experiments rather than trying to combine existing studies. This is also closely related to the issue of replication [Shull02]. Some experiences from conducting realistic experiments are presented in [Sjøberg02], in particular the article argues for funding running experiments and hiring professionals to participate in them. This is one possible way. However, we believe that we have to better understand how to transfer results from laboratory experiments with students as a complement to running larger experiments and hence also much more expensive experiments with professionals. In particular, it is a challenge to both teach empirical methods in an effective way and to simultaneously run studies that are valuable from a research perspective or from a technology transfer perspective.

When it comes to actually trying to understand the role of students as subjects in controlled experiments in software engineering, the first article addressing the topic explicitly was published in 2000 [Höst00]. The article presents an empirical study where both students and professionals participate, and then the article compares the two subject groups. In this case, no differences could be identified in performance. In another study, it is concluded that it is better to experiment within project courses than as separate exercises [Berander04]. Based on the conviction, that it is too simplistic to look solely at students vs. professionals, a scheme has been developed to include both experience and motivation [Höst05]. In another study, an interesting observation was made when a situation was found where the results from experiments with students became possible to generalize. It was a situation when the students knew one of the methods used much better but still the other method came out as being the best to use [Staron06]. When it comes to work on objects to use in experiments, the research is very limited. It is mentioned in passing in books, but no research actually addresses the challenges that have been identified.

In addition to the above, a large number of actual experiments have been published. A survey is presented in [Sjøberg05]. Furthermore, two books with a collection of empirical studies, including controlled experiments, have also been published [Juristo03, Wang03].

3. Teaching empirical methods

This section addresses issues to consider when teaching empirical software engineering on different educational levels, including bachelor, master and PhD level. Furthermore, the section discusses the use of different empirical methods in particular experiments and case studies.

3.1 Educational levels

Empirical software engineering may be taught in at least three distinct ways:

- Integration in software engineering courses
  One way of making empirical studies a natural part of evaluating and assessing different methods, techniques and tools is to integrate empirical work as assignments or studies within other courses. This means that in, for example, a course on software design different ways of performing a design can be compared and evaluated empirically. Another example is to include an experiment in a verification and validation course, where, for example, unit testing and inspection could be compared. In particular, it may be interesting to evaluate which type of

faults that are found with the different techniques. Integration into courses early in the curricula means that students get exposed to and used to apply empirical methods.

- Separate course
  It is also possible to run a separate course on empirical software engineering. The major advantage is that the course becomes very focused on empiricism and hence it becomes easier to provide assignments that include designing studies, reviewing empirical work and also possible write an empirical paper. The latter may be done without actually running a study, i.e. the focus could be in identifying, designing and discussing an empirical study. A potential drawback may be that it is hard to get a separate course on empirical software engineering into the curricula. It is often many different topics that should be covered in a software engineering program.
- Inclusion in research methodology course
  The need for a research methodology course normally becomes evident when students are approaching their thesis work, independent of whether it is a bachelor, master or PhD thesis. Thus, it may be natural to include an empirical software engineering component in a more general research methodology course. Other components may be the use of an analytical approach or the use of simulation.

The above three alternative ways of teaching empirical software engineering may be mapped to educational levels. The mapping is based on experiences at Blekinge Institute of Technology to integrate empirical software engineering into a general software engineering program. At Blekinge, students may get five different degrees in software engineering. The first three are regarded as being on undergraduate and graduate level and they are (counting years from the start at the university): 1) university diploma in software engineering after two years, 2) a bachelor degree after three years and 3) a master degree after 4.5 years. The two latter includes a thesis, where the thesis on the master level normally is more research-oriented than the thesis on bachelor level. On a research level, two different degrees are awarded: 1) licentiate degree after two years of research studies and 2) a doctoral degree after four years of research studies. The licentiate degree includes a year of course work and a thesis equivalent to one year of full time research. The doctoral degree includes 1.5 years of course work and a thesis equivalent to 2.5 years of research. The work to licentiate level is expected to be included in the doctoral degree. To simplify a little and make the Swedish system comparable to an international context, the main focus is set on the bachelor, master and PhD levels. These three levels are also the ones included in a joint agreement within the European community to ensure compatibility between countries within the community.

In our experience, it is suitable to try to integrate assignments with some empirical parts into courses on the bachelor level. It is also possible to successfully perform experimental studies with students as subjects on the bachelor level. However, it is crucial to ensure a very clear educational goal with such studies. This is particularly important on the bachelor level since the students are still quite far from research.

On the master level, it is still important to integrate empirical methods and studies into other courses. The students are now approaching the research level and in particular they are expected to write theses with a research component, and hence it is possible to run experimental studies that are more research focused. However, it is still very important to discuss the outcome of the studies and relate to whatever topic they are studying. Before the master thesis, it is suitable to run a course on either empirical software engineering or a more general research methodology course with an empirical component. The actual choice is dependent on whether the thesis work is expected to be only empirical or if other research approaches are also possible. It also matters whether the objective is to run a joint course between several programs, for example, a joint course between software engineering and computer science. The latter is the situation at Blekinge and hence methods for empirical studies are included in a more general research methodology course.

Finally, on the PhD level, we have experience from teaching specific courses on controlled experiment in software engineering, empirical software engineering and also a course on statistical methods for software engineers. The latter course is based on having data from a number of empirical studies in software engineering. The course is given using a problem-based approach. This means that the students are given a set of assignments connected to the data sets provided. No lectures are given on how to solve the problems and no literature on statistics are provided. It is up to the students to find suitable literature and hence statistical methods to use based on the assignments and the data available to them. The students practice statistical methods useful both for case study data and data from controlled experiments. Students provide individual solutions that are peer reviewed before a discussion session. In the discussion session, differences in solutions are discussed and the teacher shares her or his experience with the students. The sessions include discussions on both the statistical methods used and the interpretation of the analysis in a software engineering context. This course is run for the second time in 2006 with participants from five different Swedish universities.

3.2 Different types of studies

Most empirical studies run in a student context are controlled experiments. This poses some challenges. First of all, it is important to ensure that any study run contribute to the learning process of the students. Controlled experiments are often run with small artifacts and they are run standalone, i.e. they are not normally integrated into a development project. Given that the students in most cases will work in software projects, it is crucial to show how the knowledge gained from an experiment may be important also in the larger development context. Second, it is important that the main objective is on the educational goals. Controlled experiments may be important either from a research perspective or from a technology transfer perspective, but they may not be the most important study for the researcher if using students as subjects. A discussion on these challenges when conducting controlled experiment with students is provided in Section 4.

Experiments are most common when involving students. However, case studies should not be forgotten. They could often provide valuable information in a more project-like context. In many software engineering educations, development projects are an important part of the curricula. This does not only include individual projects. In many cases, smaller (4-6 students) projects or more large scale (12-18 students) software projects are run. These projects are not industrial projects, but they have a number of things in common with industrial projects, such as requiring that people contribute to a common goal. In some cases, these types of projects may have industrial customers. An important research question would be to identify in which cases student projects would be relevant to use in research. Once again, the research focus should not interfere with the educational goals. In [Berander04], an example is provided where it was shown that students working in projects behaved more as in industry than students participating in a controlled experiment.

4. Balancing objectives

A particular challenge is to balance different objectives when conducting empirical studies as part of the curricula. If a study is part of a course, then the research objectives should not be allowed to dominate over the educational goals. In other instances, it is not only a matter of research; it is also a matter of using an empirical study in an academic context as a stepping stone in a technology transfer process. In other words, a researcher would like to evaluate a method, technique or tool in an academic setting before moving it out to, for example, an industrial research partner. The balance between objectives puts some constraints on the actual empirical study from a research perspective:

- Clearly connected to educational goals.
- Use of students is always challenged.
- Mandatory participation may affect the results, but optional participation is not necessarily better.
- Objects in the study must be reasonable for the different perspectives.
- Comparisons between competing methods, techniques or tools must be fair.

The latter two items may require some further elaboration. Students as subjects have been discussed more in the literature than the use reasonable objects, for example, in [Höst00, Carrver03]. From an educational perspective objects in a study should not be too large due to time constraints for the students. Moreover, it may be preferable if the objects contain certain aspects or constructs that relate to what have been taught in a course. This may include, for example, certain constructs in a design method. From a research perspective, it may be preferred if the objects resemble industrial use. This may mean that objects ought to be larger, and it may also mean that certain parts of, for example, a design method should not be used or should be used. This may particularly be the case when the research is conducted as part of a technology transfer process to a specific company in which case it would be preferable to resemble that actual intended use at that specific company. This may be contradictory to the educational objectives. This requires both a delicate balance between objectives and an increased understanding of how objects should be constructed to help in balancing the objectives.

The other issue related to the objects is the fairness. It is not fair to teach one method and briefly introduce another method, and then compare them. This would clearly favor the method having been taught. Thus, it is important that methods, techniques and tools are introduced in similar ways to ensure comparability from a research perspective. There is one potentially valuable exception. If students have a thorough introduction to one out of two competing, for example, test methods and a brief introduction to the second method, and the second method comes out as the best then it is interesting. This is interesting since the opposite would normally be expected and hence it seems like the second method is not only better; it is probably superior.

In summary, too much focus is set on the use of students in research studies. There is a need to better understand the whole context of an empirical study in a student setting. The challenge is to develop our research methodology so that we better know how to gain the most knowledge from empirical studies with students. To simply disregard such studies is too simplistic and it may mean that learning opportunities are lost. As realistic studies as possible is good for both education and research. Thus, we must work with improving our understanding of how to run and how to make use of empirical studies within education.

5. Summary and conclusions

This chapter presents some ways to integrate empirical software engineering into the curricula. Some issues related to this introduction are highlighted. In particular, the need to balance educational objectives with research objectives is stressed. It is argued that both education and research would benefit from as realistic empirical studies as possible when performing studies involving students. Thus, it can be concluded that the challenge is to integrate empirical software engineering and empirical studies into the curriculum and maintain research relevance and quality. Instead of being negative towards studies with students, we should increase our understanding of how empirical studies with students can be used as part of our research process. We must address questions such as:

- How is empirical software engineering best introduced into the curricula to ensure that students are both able to run empirical studies and capable of understating their value?
- Is it possible to effectively combine educational and research objectives when performing empirical studies in a student setting?
- Can empirical studies with students be a natural stepping stone in technology transfer?

Empirical software engineering has established itself as an important area within software engineering. However, it still remains to effectively introduce it into computer science and software engineering curricula, and to address several challenges in relation to the introduction.

References

[Basili86] V. R. Basili , R. W. Selby , D. H. Hutchens, Experimentation in software engineering, IEEE Trans. on Software Engineering, 12(7):733-743, 1986.

[Basili99] V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. IEEE Trans.on Software Engineering, 25(4):456–473, 1999.

[Berander04] P. Berander. Using students as subjects in requirements prioritization. In Proc. 3rd Int. Symposium on Empirical Software Engineering, pages 167–176, 2004.

[Carver03] J. Carver. L. Jaccheri, S. Morasca and F. Shull. Issues in using students in empirical studies in software engineering education. In Proc. Int. Software Metrics Symposium, pages 239-249, 2003.

[Hayes99] W. Hayes. Research synthesis in software engineering: A case for meta-analysis. In Proc. 6th Int. Software Metrics Symposium, pages 143–151, 1999.

[Höst00] M. Höst, B. Regnell, and C. Wohlin. Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering: An International Journal, 5(3):201–214, 2000.

[Höst02] M. Höst. Introducing empirical software engineering methods in education. In Proc. Int. Conf. on Software Engineering Education and Training, pages 170-179, 2002.

[Höst05] M. Höst, C. Wohlin and T. Thelin. Experimental context classification: Incentives and experience of subjects. In Proc. Int. Conference on Software Engineering, pages 470-478, 2005.

[Juristo01] N. Juristo and A. M. Moreno. Basics of Software Engineering Experimentation. Kluwer Academic Publishers, 2001.

[Juristo03] N. Juristo and A. Moreno (editors). Lecture Notes on Empirical Software Engineering, World Scientific Publishers, 2003.

[Kitchenham04] B. A. Kitchenham, T. Dybå, and M. Jørgensen. Evidence-based software engineering. In Proc. Int. Conference on Software Engineering, pages 273–281, 2004.

[Miller99] J. Miller. Can results from software engineering experiments be safely combined? In Proc. 6th Int. Software Metrics Symposium, pages 152–158, 1999.

[Shull02] F. Shull, V. R. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonca, and S. Fabbri. Replicating software engineering experiments: Addressing the tacit knowledge problem. In Proc. 1st Int. Symposium on Empirical Software Engineering, pages 7–16, 2002.

[Sjøberg02] D. I. K. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanovic, E. F. Koren, and M. Vokác. Conducting realistic experiments in software engineering. In Proc. 1st Int. Symposium on Empirical Software Engineering, pages 17–26, 2002.

[Sjøberg05] D. I. K. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanović, N-K Liborg and A. C. Rekdal. A survey of controlled experiment in software engineering. IEEE Trans. on Software Engineering, 31(9): 733-753, 2005.

[Staron06] M. Staron, L. Kuzniarz and C. Wohlin. Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments. Journal of Systems and Software, 79(5):727-742, 2006**.**

[Thelin03] T. Thelin, P. Runeson and C. Wohlin, "An Experimental Comparison of Usage-Based and Checklist-Based Reading", IEEE Transactions on Software Engineering, 29(8):687-704, 2003.

[Wang03] A. I. Wang and R. Conradi (editors). Lecture Notes in Computer Science: Empirical Methods and Studies in Software Engineering: Experiences from ESERNET, Springer Verlag, 2003.

[Wohlin99] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in Software Engineering – An Introduction. Kluwer Academic Publishers, 1999.