

Educational Approach to an Experiment in a Software Architecture Course

Alf Inge Wang¹, Erik Arisholm^{2*}, and Letizia Jaccheri^{1*}

¹Dept. of Computer and Info. Science,
Norwegian Univ. of Science and Technology
NO-7491 Trondheim, Norway
alfw/letizia@idi.ntnu.no

²Dept. of Software Engineering,
Simula Research Laboratory
NO-1325 Lysaker, Norway
erika@simula.no

Abstract

This paper reports experiences from an experiment in a software architecture course where the focus was both on giving students valuable education as well as getting important empirical results. The paper describes how the experiment was integrated in the course, and presents an evaluation of the experiment from an educational point of view. Further, the paper reflects on the costs and the benefits of carrying out an experiment in the context of a software architecture course for the involved stakeholders namely the researchers, the students, and the instructors. The main results show that we managed to integrate the experiment in the course, that the students learned several aspects of software architecture through the experiment, and that the main motivation for the students to put effort into the experiment was to prove own skills. We also describe some guidelines for planning and executing experiments as a part of a software engineering course.

Keywords: *Software engineering education, experiment, software architecture.*

1 Introduction

Empirical software engineering (SE) gathers evidence in support of hypotheses to allow better decisions in e.g., choice of SE methods through experiments. Many of these experiments have been carried out as a part of SE courses or the subjects have been recruited from such courses. Usually, the main motivation for the experiments is to get some empirical results from using students to test various methods, tools or technologies. Students are often used in experiments for economical and practical reasons. In many cases, the educational aspects of the experiment are neglected. This might cause students to feel they have just *been used* without getting anything back. In some cases this will also ruin the empirical results of the experiment e.g., that the students lack motivation. From an educational point, it can be hard to integrate an experiment into a SE course in such way that it is integrated with the curriculum. If the experiments are "too out of place", it will become clearly for the students that the experiment is not beneficial for them (only for the researcher). In this paper, we will present experiences from integrating a controlled experiment in a software architecture (SWA) course where the focus from the beginning was both on giving students valuable education and getting important empirical results. The main contribution of this paper is an evaluation of the experiment from an educational point of view focusing on the three roles involved; researcher, student and instructor. We also investigate the main motivation for putting an effort into the tasks of the experiment. The results described is based on data from a textual debrief of the students (59 of 66 respondents) and an oral debrief of the instructors and the researchers.

* Also with Dept. of Informatics, Univ. of Oslo, PO Box 1080 Blindern, N-0316 Oslo, Norway

The general context of this work is in the intersection of SE education (especially SWA) and empirical SE [12, 14]. Our work has its roots in on a long tradition at the Norwegian University of Science and Technology (NTNU) of project-based SE education [1] and international sources [5, 10]. Bagert et al. describe a SE body of knowledge and curriculum model, and advocates for the importance of including SWA as part of SE core area [5]. Lethbridge reports about a survey that has been run among computing practitioners with the goal of assessing the relevance of their education [10]. The topic "General sw archit. & design" is the one which scores highest with respect of perceived importance, importance in career, and desire to learn more. An example of a SWA course contextualized in the education field can be found in [9].

There are several papers that describe empirical SE in an educational setting. Höst et al. assess the validity of a set of empirical studies in SE where students are used instead of professionals, and conclude that students can be used instead of professionals under certain conditions [8]. Note however, that one should be careful before generalizing such results outside of the particular study. The impact of work experience on the results of a controlled experiment clearly depends on the research question being asked [3]. Port and Klappholz discuss the benefits of student experiments not only from an experimentation point of view but also as a tool to improve the value of the provided education as students are exposed to contexts in which industry plays a significant role [11]. Carver et al. describe a similar perspective by presenting a framework for assessing student experiments from four points of view: researchers, students, instructors, and industry [7]. For each of these roles, this paper defines costs and benefits of empirical studies with students.

The paper is organised as follows. Section 2 describes experiment. Section 3 reports on the results from evaluating the experiment from an educational point of view. Section 4 discusses the impact of having an experiment in a SE from different views. Section 5 describes guidelines for experiments and SE courses. Finally, Section 6 concludes the paper.

2 The Experiment

This section describes the context, and the scientific and educational aspects of a Java programming experiment that was integrated to be a part of a SWA course at NTNU. This course is taught to 4th year master students and it demands 25% of the workload of a semester. Usually, about 60-80 students attend this course every year.

2.1 Experiment Context

The experiment at NTNU is part of a series of experiments that have been conducted to assess the impact of various object-oriented design principles and methods on maintainability.

An initial controlled experiment with 48 undergraduate and graduate students at the University of Oslo (UiO) was conducted in 1999 [4]. This experiment tested several hypotheses on how the control-style (centralized control vs. delegated control) of object-oriented designs affected the ease of performing change tasks of a Java system. The subjects performed change tasks using pen and paper on code for the control-logic of a vending machine. The results indicated that the delegated control style - despite being considered a more "proper" OO design - required much more effort to change correctly for the given sample of students.

During 2001 - 2002, a replication of the initial experiment was performed [3] where 99 junior, intermediate and senior Java consultants and 59 undergraduate and graduate students from UiO participated. Here the subjects solved the tasks using professional development tools. To manage the logistics of this experiment, the subjects used the web-based Simula Experiment Support Environment (SESE) to download code and task-descriptions, upload task solutions and answer questionnaires [2]. Compared to the previous experiment, it was more representative sample, a more realistic programming environment and use of SESE. The experimental systems and tasks were otherwise identical. The results mainly confirmed the conclusions from the initial experiment, with one important exception: Senior consultants did seem to have the necessary skills to benefit

from a delegated control style. Thus, the effect of control style on maintainability depends, to a large extent, on the skill of the developers who are going to maintain it.

In the experiment described in this paper, 66 students from NTNU participated, and their results were compared with the 59 students in the previous experiment [3], forming a quasi-experiment with 125 subjects. The experimental systems and tasks, development tools, and the use of an experiment support system to support the logistics were identical to that used in [3]. The objective of this experiment was to investigate how the maintainability of a system is affected by the order of the change tasks and how the design approach moderates the results. The subjects were divided into four groups with variations in two dimensions: design control style (centralized vs. delegated) and task order (easy-first vs. hard-first). The variation in task order was implemented as follows: The easy-first group performed the easiest change first then progressively harder (c1, c2 and c3) while the hard-first started with the hardest change task first, then progressively easier (c3, c2 and c1). For both easy-first and hard-first, the groups had to do the same change task c4 to measure the maintainability of the system (used as a benchmark). The maintainability of the system was measured by *correctness* (correct vs. incorrect solution) and *effort* (time used to make the changes). To adjust for individual skill differences, all subjects in experiment performed a pre-test programming task. Four research questions were investigated in this experiment: (1) the effect of task order on duration, (2) the moderating effect of design on duration, (3) the effect of task order on correctness, and (4) the moderating effect of design on correctness. The main results with regards to the impact of task order on maintainability are reported in [13].

2.2 The Educational Goal and Context

The educational goal of the SWA course presented in the first lecture was:

”The students should learn to define and explain central concepts in the SWA domain and learn to use and describe design patterns, methods to design SWA, methods and techniques to achieve software qualities, methods to document SWA, and methods to evaluate SWA.”

The students must learn both the theory of SWA as well as practical usage of methods and techniques. The most important aspects of the practical part of the course is to learn to use methods and techniques to achieve software qualities on different abstraction levels from low-level idioms; design patterns to high-level architectural patterns. Ultimately, the students should understand the relationships between the source code and the quality attributes of the same system.

Usually, SE courses - including SWA courses - teach students various methods and techniques that will affect the development of software in various ways. It is important to teach the students how to assess the impact of such methods and techniques in terms of e.g. time-to-market, scalability, performance, etc. A common way of assessing this impact is to use empirical studies. We therefore had a secondary goal for the course to give students an insight into empirical SE and the methods for carrying out empirical studies. The introduction of an experiment in the course was very useful in this respect.

Along with the presentation of the education goal of the course, the students got a presentation of the means to achieve the goals. The educational approaches to teach the course consisted of five main parts: *Lectures*, covering most of the curriculum; *guest lectures*, from industry on use of design patterns, performance engineering and enterprise architectures; *experiment*, carried out individually; a *SWA project*, carried out in groups of 4 students making up 40% of the grade; and a *written exam*, making up 60% of the grade. The experiment was exchanged with an exercise on design patterns that we usually run every year.

The experiment was placed in the 6th week of the course schedule after lectures covering the fundamentals of SWA, understanding quality attributes, achieving quality attributes, and architecture/design patterns. The scheduling of the experiment was intended to give the students the correct context for doing the experiment and to relate the experiment to the most recent lectures. The students did not know that the experiment was scheduled nicely according to the lecture plan,

since they did not know the contents of the experiment in beforehand. The only information given to the students was that it was a Java programming task and that it evaluated some effects. They were also told that the experiment was related to the course curriculum, that they could bring Java books and own laptops, that it was not an exam, and that we would not look at individual performance. Further they were told that they would get 1000 Norwegian Kroner (about \$150) each for doing the work and they were expected to work a maximum of 8 hours. For those students who would or could not attend, there was an alternative exercise they had to do.

One week before the experiment, the students were reminded about the experiment with practical information about when to meet, what to bring, and lunch arrangements.

2.3 Experiment Execution

In preparation for experiment day, student laboratories and a lunchroom were booked, a large order of pizza was made, and soft drinks bought. On experiment day, the students showed up on time, and the experiment started reasonably on schedule. An introduction was made to the experiment, explaining that they were a part of one of the largest controlled experiment in SE. Each was given a login id and a password to the web-based experimentation tool (SESE), and was told to start. The experimentation tool explained everything the students had to know about the experiment and collected all required information through forms and recorded time-stamps. The experiment consisted of several change tasks the students had to implement in a Java system. The students were free to leave as they finished all their tasks.

The students were sitting in two computer labs at the same floor monitored by three researchers. In addition, we had one person to manage the lunch. The students were not allowed to speak to each other or look at each other's monitors. Most students were finished after 6 hours, and only a few students had to spend all 8 hours.

After the experiment was completed, the students were asked to perform an anonymous debrief on the web by answering 13 questions. The main motivation was to reveal the attitude towards doing an experiment as a part of SE courses and to reveal the main motivation for putting an effort into the experiment. In addition, we wanted to find what the students had learned from the experiment, and positive and negative aspects of the execution of the experiment.

In the 12th week of the course, preliminary results of the experiment were presented, and the students got to know the goal and the purpose of the experiment. A Java expert had gone through all source code of the 66 students to evaluate their solutions according to correctness, extensibility, cost effectiveness, etc. In this presentation, we presented the background and the context of the experiment, the methods that were used, the experiment design, the analysis and results, and some preliminary conclusions. Finally, we presented the results from the debrief of the students and gave the students the opportunity to give oral feedback on the whole experiment experience.

3 Evaluation of the Experiment from a Educational Point of View

The evaluation is based on the data we received from the debrief of the experiment right after the experiment and the oral feedback given by student after the presentation of the experiment results.

There were two main goals for the experiment: The scientific goal to reject or confirm hypotheses based on empirical data, and the educational goal that the experiment should improve the students' knowledge and skills in the SWA course.

3.1 Integration of the Experiment and Software Architecture Course

From the data of the debrief, we first wanted to investigate how well the experiment fit in the course. On the question *"I think the experiment was relevant to the course"*, 66% agreed, 27% were neutral, and 7% disagreed¹. On the question, *"I think experiments should be used in courses"*, 60%

¹Note that the debrief was carried out before the students got to know the purpose of the experiment.

agreed, 33% were neutral, and 7% disagreed. In an open question about the benefits of participating in the experiment (in the debrief), we got a range of different responses from more pragmatically issues like good organisation and execution to issues related to the course and education. 34% of the students expressed a good feeling of mastering a programming exercise on their own. 44 % of the students reported that they discovered that there is a relationship between the structure of a system and the extendibility of the same system. Our SWA course uses the textbook "Software Architecture in Practice" [6] and the definition of SWA given in this book:

"The software architecture of a program or computing system is the structure structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [6]"

As we can see, many of the students have through the experiment discovered what a SWA is from getting to know an unknown source code and making modification to this source code. If we look back to the education goal of the course presented in Section 2.2, the experiment has contributed to achieve the first part of the educational goal "*learn to define and explain central concepts in the SWA domain*". Another statement from 9% of the students was that they finally got to understand how object orientation and structures could be used to solve practical problems. Further, that modifications to a system can cause many problems for later modifications if they are not carried out with extensibility in mind. Finally, 14% of the students expressed² that it is important to understand the SWA (see the definition above) of a system before changes can be made without causing extra problems. From the debrief, we acknowledged that the students had learned about how a SWA is reflected in the source code of the system and how the SWA affects the extensibility of the system. From previous years teaching this course we had recognized the lack of good examples on the relationship between the source code and the SWA of the system, and how the SWA reflects quality attributes. The experiment partly filled this gap. If we again look back at the educational goal in Section 2.2, the experiment has contributed in teaching the students to better use and understand "*methods and techniques to achieve software qualities*". 10% of the students acknowledged that they have improved their UML and Java skills, and learned how important it is to have a proper documentation of the system that describes how the system is structured. This statement confirms that the experiment also has contributed to an improved understanding of "*methods to document SWA*" that was also an important educational goal of the course. On the question "*I would like to do a similar experiment again*", 94% agreed while 6 % were neutral.

3.2 Students' Motivation for the Experiment

In the debrief, we also wanted to investigate the motivation for putting effort into the experiment. Figure 1 shows the results of the survey. When planning the experiment, we believed that the payment would be the main motivation for the students. However, the survey showed that the most important motivation was to *prove own programming skills* (38%). Even though we pointed out for the students that the experiment was not a programming competition, the students wanted to prove their programming skills for themselves and others. In shared second place, the students reported *do their duty* (17%) and to *provide important data to research* (17%). The students feeling that they are obliged to do their duty might explain the former. The latter might be explained by the information we gave the students that the experiment was a part of a very big and important SE experiment. 14% of the students said that to *learn something* was the main motivation, and most likely to learn more about SWA through the experiment. A bit surprising, the *payment* was the main motivation for only 10% of the students. It would have been interesting to see how many students would have participated in the experiment if we had not provided any payment and how this would have affected the motivation and the performance of the students. Finally, 4% of the students reported *other reasons* (not specified) as the main motivation for the experiment performance.

During the semester, the course staff monitored how the students perceived the SWA course in terms of 1) relevant for practice, 2) usefulness after studies, 3) interesting topic and 4) importance.

²Answer from an open question: What did you learn from the experiment?

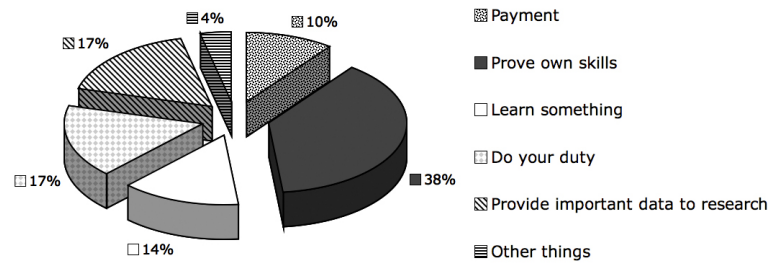


Figure 1. The most important motivation to put real effort into the experiment

This information was collected through forms the students had to fill out 4 times in the semester. The results from the forms indicated that the experiment gave a boost in two areas: 1) relevant for practice and 4) importance.

4 Discussion

The discussion in this section is inspired by [7], reflecting cost/benefit from three different viewpoints.

4.1 Cost/Benefit for the Researcher

A possible cost for the researcher is the threats to validity introduced by student experiments. A subject sample consisting of students may not be representative of the intended target population, which usually would be professional developers. The degree to which results from student experiments can be generalized to professionals is thus uncertain. For example, the conclusions from the control-style experiment described in Section 2.1 differed significantly depending on the level of experience of the subjects [3]. However, for tasks involving the use of new technologies, students might in fact be better trained than most professionals. In those cases, although the students are still not representative of the current population of professionals, they might be the best surrogates available for the next generation of professionals.

Usually, the main motivation for conducting experiments as a part of a SE course is for economical and practical reasons. However, unlike the similar experiment using professional Java consultants [3], it is possible to control the working environment providing homogeneous computer environment and tools. By using computer-labs instead of own offices, it was easier to control and monitor the behaviour of the subjects during the experiment, and clarify upcoming problems that can affect the scientific results.

Another benefit from conducting experiments as a part of a course is the opportunity to train junior researchers. In our experiment, the course staff had little experience in conducting empirical studies and was fortunate to get the necessary training, guidance and help from an experienced researcher. This training makes it possible for the course staff to conduct experiments also in other courses. The experiences, knowledge and tools from this experiment can be used as a basis for planning and executing more experiments as parts of software engineering courses in the future.

4.2 Cost/Benefit for the Student

When introducing an experiment in the SWA course, the focus was on minimizing the cost and maximizing the benefits for the students. The total lost education time was about 1 week of lectures and 8 hours of exercises. 30 min were used to introduce the experiment, and 2x45 min to present the results. The loss of 8 hours of the exercise time caused by that we had to drop a pattern exercise for

students participating in the experiment. From the debrief of the experiment, 15% of the students expressed that they did not learn anything from participating in the experiment³.

The main benefits for students were improved self confidence as programmers, improved programming skills, engaging way to work and learn, and improved SWA understanding. One student expressed that this was his first work-relevant exercise in his 4 years of studying.

4.3 Cost/Benefit for the Instructor

The role of the instructor is very different from the role of the researcher in an experiment, even if the same person plays both roles. For the instructor, it is important to achieve the educational goals and that the scientific goals do not put obstacles in the way of the former. The cost of having an experiment for the instructor was in our case the loss of lecturing time and the additional time required to get acquainted with the experiment. Integrating the experiment to fit well with the contents and course schedule minimized this cost.

From the instructor's point of view, we discovered many benefits of introducing an experiment into the course. By going through the results of the programming tasks in the experiment, the instructor got a better view of the programming capabilities of the students. The introduction of the experiment gave a boost in the students' morale and motivation. We also noticed that the experiment improved to keep the students' attention towards the course, i.e., students frequently asked in lectures what was the purpose of the experiment and if the results were ready soon. An empirical SE paper was integrated as a part of the course material [3] to improve the understanding of the experiment and empirical methods. Finally, the experiment helped to improve the programming skills of the students. It is no use teaching students SE, if they do not know how to understand, develop and modify programs.

5 Guidelines for experiments in Software Engineering Courses

The guidelines for integrating experiments in SE courses is based on experiences from the experiment described in this paper, several experiments in previous courses and [7]:

- G1 **Plan educational aspects carefully:** It is usually obvious to carefully plan the experiment from an experimental point-of-view. However, it is also important to plan the educational aspects of the experiment. The most important issues include to schedule the experiment well according to the course plan and to have the experiment results ready within the semester (as soon as possible).
- G2 **Make sure the topic of the experiment fit the course:** Our worst experiences with experiments are where the students have boycotted because the experiment was too out of place. Make sure to identify a strong link between the course and the experiment.
- G3 **Brief and debrief the experiment:** The students need to be motivated before the experiment (without revealing any details), and give course related feedback based on the experiment results.
- G4 **Use familiar work environment whenever possible:** The experiment results will improve if the students work with familiar tools and in the computer labs they ordinary use.

6 Conclusion

Based on the experiences presented in this paper, we will argue that it is very important to have both scientific and educational goals when incorporating experiments in SE courses. As a result, an experiment can improve the course by giving the students a unique learning experience and thus

³Note that the debrief was conducted before they got a presentation of the experiment results.

boosting the morale in the class as well as improving the experiment results by improved motivation. This is only possible if the experiment goals intersects the educational goal of the course. If the experiment is not related to the course at all, a negative boost in morale might ruin the results.

There is always room for improvements, and we have acknowledged some issues that could have improved the educational aspects of the experiment implementation. Firstly, the time between the experiment and the feedback of the experiment should be reduced to improve the educational effects. In our case, it was 6 weeks and this was a bit too long. Secondly, it would have been beneficial from a SWA point of view if the system to be modified had been a more complex system than a vending machine. Thirdly, it would also been beneficial to see the effect of more quality attributes than *extensibility* in the experiment.

References

- [1] Rudolf Andersen, Reidar Conradi, John Krogstie, Guttorm Sindre, and Arne Sølvsberg. Project Courses at the NTH: 20 years of Experience. In *J. L. Diaz-Herrera (ed.): "7th Conference on Software Engineering Education (CSEE'7)"*, pages 177–188, San Antonio, USA, 5–7 Jan. 1994. Springer Verlag LNCS 750.
- [2] E. Arisholm, D. Sjøberg, G.J. Carelius, and Y. Lindsjørn. A Web-based Support Environment for Software Engineering Experiments. *Nordic Journal of Computing*, 9(4):231–247, 2002.
- [3] Erik Arisholm and Dag Sjøberg. Evaluating the Effect of a Delegated versus Centralized Control Style on the Maintainability of Object-Oriented Software. *IEEE Transactions on Software Engineering*, 30(8):521–534, 2004.
- [4] Erik Arisholm, Dag Sjøberg, and Magne Jørgensen. Assessing the Changeability of two Object-Oriented Design Alternatives - a Controlled Experiment. *Empirical Software Engineering*, 6(3):231–277, 2001.
- [5] D. Bagert, T. Hilburn, G. Hislop, M. Lutz, M. McCracken, and S. Mengel. Guidelines for software engineering education, 1999.
- [6] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice - Second Edition*. Sei series in Software Engineering. Addison-Wesley, 2003.
- [7] Jeffrey Carver, Letizia Jaccheri, Sandro Morasca, and Forrest Shull. Issues in Using Students in Empirical Studies in Software Engineering Education. In *9th International Software Metrics Symposium*, pages 265–272, Sydney, Australia, September 3–5 2003.
- [8] Martin Höst, Björn Regnell, and Claes Wohlin. Using students as subjects a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, 2000.
- [9] Patricia Lago and Hans van Vliet. Teaching a course in Software Architecture. In T.C. Lethbridge and D. Port, editors, *18th Conference on Software Engineering Education & Training (Eds. T.C. Lethbridge and D. Port)*, pages 35–42. IEEE Computer Society Press, 2005.
- [10] Timothy C. Lethbridge. The relevance of software education: A survey and some recommendations. *Annals of Software Engineering*, 6:91–110, 1998.
- [11] Daniel Port and David Klappholz. Empirical research in the software engineering classroom. In *CSEE&T*, pages 132–137. IEEE Computer Society, 2004.
- [12] Walter F. Tichy. Should Computer Scientists Experiment More? *IEEE Computer*, 31(5):32–40, May 1998.
- [13] A.I. Wang and E. Arisholm. The Effect of Task Order on the Maintainability of Object-Oriented Software. Technical report, Norwegian University of Science and Technology, 2007. IDI-TR-02-07, Available on Web: <http://www.idi.ntnu.no/grupper/su/publ/alfw/idi-tr-02-07.pdf>.
- [14] Marvin V. Zelkowitz and Dolores R. Wallace. Experimental Models for Validating Technology. *IEEE Computer*, 31(5):23–31, May 1998.