Abstract: An important goal of most empirical software engineering experiments is the transfer of the research results to industrial applications. To convince industry about the validity and applicability of the results of controlled software engineering experiments, the tasks, subjects and the environments should be as realistic as practically possible. Such experiments are, however, more demanding and expensive than experiments involving students, small tasks and pen-and-paper environments. This article describes challenges of increasing the realism of controlled experiments and lessons learned from the experiments that have been conducted at Simula Research Laboratory.

Keywords: empirical software engineering, technology transfer, controlled experiments

5.1 Introduction

The ultimate criterion for success in an applied discipline such as software engineering (SE) research is the widespread adoption of research results into everyday industrial practice. To achieve this, diffusion of innovation models requires the evidential credibility of software experiments, which depends on both the producer and the receiver of the results. Without a close tie between the experimental situation and the “real”, industrial situation, practitioners may perceive the experiment as irrelevant and ignore the results.

Hence, there is an increasing understanding in the software engineering community that realistic empirical studies are needed to develop or improve processes, methods and tools for software development and maintenance [1, 2, 3, 4, 5, 6, 7].

Most of the studies in software engineering that have emphasized realism are case studies. However, a major deficiency of case studies is that many variables vary from one case study to another so that comparing the results to detect cause-effect
relationships is difficult [8]. Therefore, controlled experiments should be conducted to complement case studies in empirical software engineering.

While the raison d’être for experimental research is to establish evidence for causality through internal logical rigor and control [9], this is not enough. It is also important to ensure external validity [8]. If an experiment lacks external validity, its findings may not be true beyond the experimental setting. An important issue, therefore, is whether the particular features of formal SE experiments are realistic. In particular, it is a challenge to achieve realism regarding experimental subjects, tasks and environment [10].

Controlled experiments in software engineering often involve students solving small pen-and-paper tasks in a classroom setting. A major criticism of such experiments is their lack of realism [11, 12], which may deter technology transfer from the research community to industry. The experiments would be more realistic if they are run on realistic tasks on realistic systems with professionals using their usual development technology in their usual workplace environment [13]. Generally, a weakness of most software engineering research is that one is rarely explicit about the target population regarding tasks, subjects and environments.

During the last couple of years the authors of this chapter have conducted 13 controlled experiments with a total of 800 students and 300 professionals as subjects. The purpose of this article is to describe some of the challenges and risks to be encountered when the realism of controlled experiments increases, and to give some recommendations based on our experience.

The remainder of this article is organised as follows. Sections 5.2–5.4 discuss challenges and lessons learned from our attempt of increasing the realism of respectively subjects, tasks and environment of controlled SE experiments. Section 5.5 addresses the logistics of conducting such experiments. Section 5.6 concludes.

### 5.2 Representative Subjects

A prerequisite for the discussion on realism is that we are conscious about the population we wish to make claims about [14]. Implicit in our discussion is that the interesting population is “representative” software builders doing “representative” tasks in “representative” industrial environments. Nevertheless, it is not trivial defining what “representative” means. For example, there may be many categories of professionals, such as junior, intermediate and senior consultants.

#### 5.2.1 Target and Sample Population

As in all experimental disciplines involving people, two major challenges are:

- Identifying the population about which we wish to make claims and
- Selecting subjects who are representative of that population.
The similarity of the subjects of an experiment to the people who will use the technology impacts the ease of the technology transfer [15]. Unfortunately, few papers reporting controlled SE experiments are explicit on the target population. Usually, there is an assumption that the target population is “professional software builders”. One should; however, be aware that this group may be very diverse. That is, the performance may differ significantly between various categories of professionals [16].

A common criticism of experiments in software engineering is that most of the subjects are students, which might make it difficult to generalise the results to settings with various kinds of professionals. To simplify the generalisation of experimental results to a realistic setting, one should attempt to sample subjects from the population of professionals that we wish to make claims about. However, students are more accessible and easier to organise, and hiring them is generally inexpensive. Consequently, experiments with students are easier to run than experiments with professionals and the risks are lower.

Student experiments should thus be used to test experimental design and initial hypotheses, before conducting experiments with professionals [17]. Experiments with students might have the goal of gaining an understanding of the basic issues without actually aiming for external validity. Conducting “unrealistic” experiments may be a first step in a technology transfer process, that is, to reduce risks and costs, one should start with a relatively small experiment with students, possibly with small tasks (see Sect. 5.3.2) and the use of pen and paper (see Sect. 5.4.1), and then increase the scale of realism if the first pilot experiments are promising.

5.2.2 Background Information about Subjects

Generally, papers describing SE experiments that involve professionals often do not characterise the professionals’ competence, experience and educational background, and the authors seldom justify to what extent their subjects are representative of the software engineers who usually perform such tasks. This leads to several problems:

- The results may not be trustworthy, that is, the professionals may not be realistic for the actual experimental tasks. The sample recruited may be biased in some way, for example, a company may only be willing to let the software engineers who are least experienced or least in demand take part in an experiment.
- Comparing results from the original with replicated studies is difficult.
- Successful transfer of the results into industrial practice is less likely.

To generalise from experiments with a given group of subjects, we would need information about the ability and the variations among the subjects and the group of people to which the results will be generalised [18]. For professionals, depending on what we wish to study, it would be relevant to know the variations regarding competence, productivity, education, experience (including domains), age, culture/nationality (?), etc. (Some of this information may be highly controversial and should be carefully considered from an ethical point of view.) A challenge of
measuring these attributes is to define good measures that can be used in practice. For example, how do we measure competence and productivity? In practice, we would have to find meaningful substitute measures for those we cannot measure directly. In an experiment on object-oriented (OO) design principles [16], we collected detailed information about:

- Age
- Education (number of credits in general, number of credits in computer science)
- General work experience
- Programming experience (OO in general, particular programming languages (Java, C++, etc.)
- Knowledge of systems developments methods and tools, and
- Subjective description of their own programming skills.

This background information can be used in several ways, for example, to determine:

- The target population for which the results are valid, and
- To what extent the results of the treatments depend on the collected background information, e.g., that certain design principles might be easier to understand for experienced professionals than for novices.

It would also be interesting to identify the variations within the same company versus among companies, variations between in-house professionals versus consultants, etc. For example, in-house software development in Nokia, Ericsson, Bosch, etc. may differ from development projects run by consultancy companies. Nevertheless, knowledge about the effect of a certain technology among consultants or even students may still be useful in the lack of knowledge of the effect of the technology in a company’s own environment.

5.2.3 How to Get Subjects?

Both students and professionals are used in SE experiments. For most university researchers it is relatively easy to use students as subjects in experiments. One can organise an experiment as follows:

1. The experiment is considered a compulsory part of a course, either as part of the teaching or as an exercise [19, 20].
2. The experiment is not compulsory; it is voluntary, but is still regarded relevant for the exam [21]. (In practice, students may feel obliged to take part to show their teacher that they are enthusiastic students.)
3. The students are paid, that is, the experiment is not considered as part of the course (but it may still be relevant) [22, 23, 24].
In our research group, we have experienced that the organisation indicated in alternative (3) is usually the easiest one. We then do not have the time constraint of the ordinary classes, the students are motivated and there seems to be few ethical problems [25, 26]. (One might argue that it is unethical if some students have been using a technology that proved better than the technologies being used by others students, that is, some students have learned better technologies than other students. However, when the experiment is voluntary and they are paid, this is hardly a problem). In any case, if practically possible, we inform the students about the results of the experiments.

The lack of professionals in software engineering experiments is due to the conception of high costs and large organisational effort. Harrison [10] puts it this way:

Professional programmers are hard to come by and are very expensive. Thus, any study that uses more than a few professional programmers must be very well funded. Further, it is difficult to come by an adequate pool of professional developers in locations that do not have a significant software development industrial base. Even if we can somehow gather a sufficiently large group of professionals, the logistics of organizing the group into a set of experimental subjects can be daunting due to schedule and location issues.

Fenton claims that “generally, such [formal] experiments are prohibitively expensive and technically infeasible to set up properly” [27]. He then refers to an experiment conducted with MSc students that was criticised for the claim that these students were representative of trainee programmers in industry.

To alleviate these problems, we have applied alternative incentives to conduct experiments with professionals:

− Offer the organisation tailored internal courses and, for example, use the course exercises as experiments
− Have a part time job in the company and advocate the experiment as useful for the company [28]
− Involve some of the employees in the research and offer them co-authorship of the resulting research paper
− Offer the organisation a network of people from other organisations with relevant experience
− Pay the company directly for the hours spent on the experiment [29]

The first and the last alternative have proved most successful. Regarding the last alternative, we thought that it would be most effective to use our personal network to get people to take part in our experiments on their spare time and pay them individually. However, it turned out that a much better approach is to phone the switchboard of a major consultancy company and request a specific service.

This way, one can get a sample of different categories of professionals (junior, intermediate, senior) from different companies as subjects in one experiment. In the OO design experiment [16], 130 professionals from nine companies took part.
They were easy to get. It was considerably more difficult to get subjects to take part in another experiment on design pattern [30], because that experiment was held for a period of three given days, whereas the length of the OO design experiment was only one day, and the day was chosen by the participating companies themselves (within a certain interval).

When we hire people from consultancy companies to take part in our experiments, we are treated professionally like any ordinary customer (although several consultants say that they find our experiments more exciting than most other projects). We agree on a contract and they internally define a project with a project leader, budget, etc. Of course, one must have the resources to do research this way.

### 5.3 Realistic Tasks

When conducting controlled experiments in software engineering, one should consider the realism and representativeness of the tasks regarding the size, complexity and duration of the involved tasks. Specification, implementation and verification methods also vary considerably between domains, such as accounting software versus flight-control systems. In our opinion, some experimental tasks bear little resemblance to actual tasks in software engineering; others are very similar to actual tasks [31]. In between there is a continuum. Larger development tasks may take months, while many maintenance tasks may take only a couple of hours.

Most experiments in software engineering seem simplified and short-term: “the experimental variable must yield an observable effect in a matter of hours rather than six months or a year” [10]. Such experiments are hardly realistic given the tasks of building and maintaining real, industrial software, particularly since many of the factors we wish to study require significant time before we can obtain meaningful results.

#### 5.3.1 Collecting Information about “Typical” Tasks

A systematic way to define representative tasks according to a given application area in a given context, is to collect information about the kinds and frequencies of tasks in the actual environment and then create “benchmark tasks”, i.e., a set of tasks that is a representative sample of tasks from the population of all tasks. An example use of such benchmark tasks is described in [32]. In that study, the maintenance benchmark tasks were derived from another study of 109 randomly sampled maintenance tasks [33].

In yet another study, we collected information about all the maintenance tasks in a tool vendor company through a Web interface during a period of six months [34].
5.3.2 Longer Experiments

Generally, to increase the realism of SE experiments, the duration of the experimental tasks should be increased. As far as we have observed, the tasks carried out in student experiments take only up to three-four hours – most of them are shorter to fit with the time schedule of a university class. In the experiment on object-oriented (OO) design principles [16], the subjects spent one day each on five experimental tasks; whereas in the design pattern experiment [30] the subjects spent three days (including a course on design patterns the second day).

We have conducted one longer-term (35 h), one-subject explorative study [35], that is, an “N=1 Experiment” [10]. The longer duration of this study allowed a wider spectrum of tasks to be carried out. The system on which the tasks were performed was also larger in size and complexity than usual in most experiments. Another positive effect of the longer duration was that the pressure from the experimental situation put on the subject was less, that is, more realistic, than what we have experienced in the controlled experiments we have run. In the student experiments, most students felt as if they were in an exam situation. “How did I do?” they asked after the experiment.

Another example of an experiment with high realism of tasks is our ongoing study on uncertainty in the estimation of development effort. In that experiment we pay an organization to evaluate three estimation processes. The organization now estimates one third of their incoming projects respectively according to the first, second and third estimation process.

Increasing the duration of the experiments enables more realistic tasks to be carried out. We have tried several means to achieve longer experiments; some of them with success (see Sect. 5.4.2). Of course, our tasks may still be small compared with many actual tasks. We are therefore planning an experiment where an application system that is actually needed by Simula Research Laboratory, will be developed by 4–5 different project teams (each consisting of 4–5 persons) from different consultancy companies. It should be possible to develop the system in a couple of months. This would then be a very realistic development task. Of course, the number of subjects (here teams) is too small to conduct hypothesis testing, but we still have some control. Nevertheless, there are many challenges to such an experiment.

5.3.3 Methodological Challenges Regarding Quality and Time

Time is often a dependent variable in SE experiments. Usually, we want the subjects to solve the tasks with satisfactory quality in as short time as possible, as most software engineering jobs put a relatively high pressure on the tasks to be done. However, if the time pressure put on the participatory subjects is too high, then the task solution quality may be reduced to the point where it becomes meaningless to use the corresponding task times in subsequent statistical analyses. A challenge is therefore to put a realistic time pressure on the subjects. How to best deal with this challenge depends to some extent on the size, duration and
location of the experiment. For smaller experiments where the subjects are located in the same physical location (e.g., a classroom), we have applied the following strategies:

- All subjects receive a fixed honorarium for their participation. This eliminates potential problems of subjects speculating in working slowly to get higher payment.
- The subjects work for the same amount of time (e.g., 3 h), finishing as many tasks as they can. This prevents faster subjects from disturbing the slower subjects. At the finishing time, everyone has to leave.
- The subjects are informed that they are not all given the same tasks. This (combined with the fixed time on the tasks) reduces the chances that the “slow” subjects deliver solutions with inadequate quality (faster than they should) in an attempt to appear smart in front of their peers; most persons would find it embarrassing to be the last to complete the tasks.
- The last task of the experiment is a large task that we a priori do not expect the subjects will be able to complete. This assumption should be tested in a small pilot experiment. Unknown to the subjects, this extra task is not included in the analysis. The extra task puts sufficient time pressure also on the fast subjects. It also reduces threats to validity caused by “ceiling effects”, that is, the adverse effect of having too much or too little time towards the end of the experiment, since this extra task is not included in the analysis.

We applied all of these strategies in the pen-and-paper version of the OO design experiment [16]. Since nobody finished the extra task, there was sufficient time pressure. Everyone left at the same time (after three hours). Hence, there was no disturbance and it was fair that everybody received a fixed honorarium. Furthermore, there was no reason to speculate in working slowly (to increase their payment), or to work faster than they should (to look good).

In our more work-intensive experiments, the tasks would typically take one day or more to complete. Furthermore, the experiment may be located in different physical locations, e.g., in their usual work environment (Sect. 5.4). In these cases, the fixed honorarium and fixed time strategies seem less appropriate since many subjects will have to be present without doing any sensible work for a longer time and disturbance is less of an issue. In these cases we have applied the following strategies:

- Instead of a “fixed” honorarium, we estimate the work to (say) 5 h, and then say that the subjects will be paid for those 5 h independently of how long they would actually need. (Note that we wish the subjects to finish as soon as possible; we would discourage people to speculate in working slowly to get higher payment.) Hence, the subjects who finish early (e.g., 2 h) are still paid for 5 h. However, in practice, we tell the subjects when the 5 h have passed, that they will be paid for additional hours if they finish their tasks.
- The subjects are allowed to leave when they finish.
Challenges and Recommendations

- As for fixed time, smaller scale “classroom” experiments, the subjects are still informed that they are not all given the same tasks to reduce the chances that they for competitive reasons work “faster” than they should with resulting low quality of the delivered task solutions.

- As for fixed time, smaller scale “classroom” experiments, the experiment should preferably still include an extra, last task not to be included in the analysis. Although the benefit of an extra task is probably not as large as for fixed time, fixed honorarium experiments, our results suggest that the last task nevertheless may exhibit ceiling effects and therefore should not be included in the analysis. The potential benefits of an extra, last task may justify the added duration and costs of the experiment.

We applied these alternative strategies for the professionals participating in the replicated OO design experiment [16]. Our experiences suggest that these strategies work fairly well, although each strategy provides different advantages and disadvantages in terms of threats to validity, practical issues and costs. For example, restrictions imposed by an existing experiment design might make it difficult to include an “extra task”, like the experiment reported in [30].

In another experiment on UML design processes with 53 students [36], we combined the strategies described above. The experiment was run in a classroom setting and was scheduled for three hours. However, due to the need for extra time caused by the use of a CASE tool (see Sect. 5.4.1); many students had not finished after the three hours. Those students were then encouraged to stay longer and finish their tasks by being offered additional payment. This way, we managed to collect more data points than if everybody had left after the scheduled time.

5.4 Realistic Environment

While our focus is on controlled experiments, this does not mean that we are only concerned with laboratory, or in vitro, experiments. Controlled experiments can also be conducted in vivo, in a more realistic environment than is possible in the artificial, sanitized laboratory situation [3]. However, the realistic environment can also be a weakness, because it may be too costly or impossible to manipulate an independent variable or to randomize treatments in real life. Thus, the amount of control varies through a continuum, and prioritizing between the validity types is an optimization problem, given the purpose of the experiment. Nevertheless, external validity is always of extreme importance whenever we wish to generalize from behaviour observed in the laboratory to behaviour outside the laboratory, or when we wish to generalize from one non-laboratory situation to another non-laboratory situation.
5.4.1 System Development Tools

Even when realistic subjects perform realistic tasks, the tasks may be carried out in an unrealistic manner. The challenge is to configure the experimental environment with an infrastructure of supporting technology (processes, methods, tools, etc.) that resembles an industrial development environment. Traditional pen-and-paper based exercises used in a classroom setting are hardly realistic for dealing with relevant problems of the size and complexity of most contemporary software systems. Recently, we have replicated three experiments where we have replaced pen and paper with professional system development tools:

− In the OO design principle experiment [16], a variety of Java development environments were used (JBuilder, Forte, Visual Age, Visual J++, Visual Café, etc.). This is a replication of the experiment described in [23].
− In the OO design pattern experiment [30], a C++ environment was used. This is a replication of the experiment described in [37].
− In the experiment on UML design processes [36], 27 subjects used a commercially available OO development CASE tool (Tau UML Suite), while 26 subjects used pen and paper. This is a replication of the experiment described in [38].

Our experience from these experiments is that using system development tools requires proper preparation:

− Licences, installations, access rights, etc. must be checked
− The subjects must be or become familiar with the tools
− The tools must be checked to demonstrate acceptable performance and stability when many subjects are working simultaneously. In one experiment, several subjects had to give up because the tool crashed.

Other researchers who plan experiments using professional tools should take into account that our results show that the time spent to solve the same tasks took 20–30% longer when using tools than when using pen and paper [16, 30, 36]. Note also that the variance also increases considerably when tools are used. This may influence the time allocated to the experiment.

Regarding quality, there are, as expected, fewer syntactical errors when tools are used. More surprising is that there seem to be more logical errors. More analysis is needed to investigate this issue into further depth. In particular, the relationships among the three realism dimensions (subjects, tasks and environment) need to be investigated, for example, regarding scalability: a professional development tool will probably become more useful the larger and more complex the tasks and application system become.
5.4.2 Experimental Procedure in a Realistic Environment

Many threats to external validity are caused by the artificial setting of the experiment. For example, because the logistics is simpler, a classroom is used instead of the usual work place. Conducting an experiment on the usual work site with professional development tools implies less control of the experiment than we would have in a classroom setting with pen and paper. Thus, there are many challenges when conducting experiments with professionals in industry. We have learned the following lessons:

- Ask for a local project manager of the company who should select subjects according to the specification of the researchers, ensure that the subjects actually turn up, ensure that the necessary tools are installed on the PCs, and carry out all other logistics, accounting, etc.
- Motivate the experiment up-front: inform the subjects about the purpose of the experiment (at a general level) and the procedure (when to take lunch or breaks, that phone calls and other interruptions should be avoided, etc.).
- Ensure that the subjects do not talk with one another in breaks, lunch, etc.
- Assure the subjects that the information about their performance is kept confidential (both within company and outside).
- Assure the company that its general performance is kept confidential.
- Monitor the experiment, that is, be visible and accessible for questions.
- Give all the subjects a small training exercise to ensure that the PC and tool environment are working properly.
- Assure the company and subjects that they will be informed about the results of the experiment (and do it).
- Provide a proper experiment support environment to help set up and monitor the experiment, and collect and manage the experimental data (see Sect. 5.2).

5.5 Supporting the Logistics of Controlled Experiments

Our experience from the experiments we have run with both students and professionals is that all the logistics around the experiments is work intensive and error prone: general information and specific task documents must be printed and distributed, personal information (bank account, etc.) and background information must be collected, all solution documents must be collected and then punched into an electronic form, etc. This may in turn lead to typing errors, lost data [39], and other problems.

We realised that if we were to scale up our experiments, and particularly run experiments with professionals in industry using professional development tools, that is, make our experiments more realistic, we would need electronic tool support. Hence, we searched for suitable tools and found several Web tools developed
to support surveys, most of them designed by psychologists (e-Experiment\(^1\), PsychExperiments\(^2\), Survey Pro \(^3\), S-Ware WWW Survey Assistant\(^4\), Wextor\(^5\)). Those tools basically distribute questionnaires to the respondents who fill them in online. Then the results are stored in a local database or sent via emails to the researchers. However, to conduct the kind of experiments that we were interested in, we needed a more sophisticated tool. Therefore, in collaboration with a software company that develops solutions for Human Resource Management, we developed (and are still extending and improving) the Web-based Simula Experiment Support Environment (SESE). SESE is built on top of the company’s standard commercial human resource management system. Fig. 5.1 illustrates the way SESE supports an experiment:

**Step 1**: The researcher defines a new experiment (SESE can manage an arbitrary number of experiments simultaneously) with the required questionnaires, task descriptions, files to be downloaded etc.

**Step 2**: The administrator creates a user-id and password for each person that will take part in the experiment, and emails that information to the person.

---

2. [http://www.olemiss.edu/PsychExps/](http://www.olemiss.edu/PsychExps/).
4. [http://or.psychology.dal.ca/~wcs/hidden/home.html](http://or.psychology.dal.ca/~wcs/hidden/home.html).
**Step 3:** The user (subject) fills in questionnaires (personal and background information) and downloads task descriptions and other required documents (design models, source code, etc.).

**Step 4:** The user carries out the tasks, answers questions along the way and uploads the finished documents. Timestamping is done continuously (when were the task descriptions downloaded and task solutions uploaded, when a break started and stopped, etc.).

**Step 5:** When a subject has finished the tasks, his or her results are stored in the (relational) database of SESE. When all the subjects have finished, the researcher can start analysing the data.

The OO design experiment was run at 10 different sites using SESE via the Web. The experiences from using SESE are positive. SESE enables us to run distributed experiments – both in location and time – instead of only “big-bang” experiments. If acceptable from a methodological point of view, one should avoid “big-bang” experiments to reduce risks. For example, in our design pattern experiment, a fibre cable breakdown far beyond our control forced us to send 44 consultants home and defer the experiment to start on the next day. This accident caused a lot of frustration and a direct loss of 20,000 €. Also in the UML experiment we had serious problems. The day before the experiment, there was a hacker break-in into the computer network of Oslo University where the experiment was to be run. All the 52,000 passwords of the university had to be changed by the systems department and all accounts were closed at the exact same time as our experiment was supposed to start. Fortunately, we managed to get a deal with the systems department to treat the accounts of our subjects as special cases.

Future extensions of SESE may include detailed logging of the way a task is performed or a technology is used. This may include window operations, keystrokes, mouse operations and movements logged with timestamps [40]. SESE and the experiences from using it are more fully described in [41].

### 5.6 Summary

This article focused on the need for conducting more realistic experiments in software engineering. Using a large experiment on OO design alternatives and other experiments conducted by our research group as examples, we described how increased realism can be achieved, particularly along the dimensions subjects, tasks and environment. A Web-based experiment supporting tool was also described briefly.

We discussed several extra challenges and larger risks that must be taken into account when conducting more realistic experiments. Based on our experiences, we described lessons learned and recommendations for tackling the challenges and reducing the risks, amongst others:

- Be explicit about your target and sample population, possibly divided into sub-populations (e.g., junior, intermediate and senior consultants).
− Record background information about subjects.
− Using professional system development tools in an experiment increases the realism, but requires careful planning, risk analysis and more resources.
− To get subjects to take part in experiments, consider the various kinds of “award” proposed.
− To help tackle the tradeoffs between quality of the tasks to be conducted in an experiment and the time spent on solving them, consider the proposed techniques.
− Apply the described guidelines on the practical conduct of experiments in industry.

We believe that many of the challenges described in this paper also are faced by other researchers conducting controlled software engineering experiments. To increase the knowledge of the empirical software engineering community in this area, we hope that more experiences on how to tackle these challenges will be reported in the literature.

References

Challenges and Recommendations


Software Process Improvement
Results and Experience from the Field
2006, XII, 414 p. 42 illus., Hardcover
ISBN: 978-3-540-32178-1