1. INTRODUCTION

Frequent changes in technology and continuously increasing demands from software customers means that software development is largely a creative process. Nevertheless, a fundamental question for software engineering in general, and for empirical software engineering in particular, is the reproducibility of software development: which aspects of a software development process and product are reproducible and which parts depend on the creativity of developers? A priori, it is not obvious that such separation into creative and reproducible parts can be made, but if that were possible, we could focus software process, technologies, and empirical studies on the reproducible parts without restricting the creativity of developers.

Therefore, this question is worthy of extensive investigation.

The precise definition of reproducibility varies across disciplines, but it is often closely related to the definitions of repeatability. In an encyclopaedia of philosophy of science [Tetens 2004], reproducibility is described as the repeatability of the process of establishing a fact, or of the conditions under which the same fact can be observed. In natural science, reproducibility is often related closely to the repeatability of experimental conditions and results. See Anda et al. [2009] for a more comprehensive discussion of the concept of reproducibility in software engineering.

The question of reproducibility can be asked for various levels of project granularity: from low-level atomic activities of developer actions to outcomes of entire projects, and for small elements of software such as a class or a function to large-scale software systems such as middleware in apache.org, Gnome UI, or even entire ecosystem of applications tied to the MS Windows platform, or event to entire Internet.

The question of reproducibility is important for practitioners because it underlies the fundamental questions that have had the attention of software researchers for years, such as, to what extent software development effort and software quality can be predicted. Furthermore, this question is important from a scientific point of view since an understanding of what it means for a phenomenon to be reproducible is a prerequisite if software engineering (SE) phenomena are to be investigated using the scientific method and, hence, if SE is to become a mature industry. Scientific inquiry and mature engineering industries are recognized by reproducibility.

The predominant focus in the empirical software engineering literature on research methods in general and replications in particular is experiment, see, for example [Wohlin et al. 1999] [Miller 2005] or [Shull et al. 2008]. Although the classical method for identifying cause-effect relationships is to conduct controlled experiments where only a few variables vary, a common criticism of SE experiments is their lack of realism, which may deter the transfer of technology from the research community to industry. Hence, a challenge is to increase the realism while retaining a relatively high likelihood that the technology under study (the treatment) is actually the cause of the observed outcome. Even though we can increase the realism of experiments, there remain many SE phenomena that occur in complex, real-life environments that cannot be studied fully through experiments, for example whole software projects. In many situations the boundaries between phenomenon under study, which in empirical software engineering studies is typically a development method or technique, and its context, typically organisation, people and tools, are not clearly evident. The results of such studies have little validity unless we can control these context variables or understand their actual moderating effect.

While an experiment deliberately divorces a phenomenon from its context, the case study aims deliberately at covering the contextual conditions. Consequently, we need case studies in software engineering [Sjoberg et al. 2008]. According to Yin [2003] a case study should be conducted “when a ‘how’ or ‘why’ question is being asked about a contemporary set of events” that are investigated within its real-life context.

The typical situation in case studies is that there are more variables of interest than data points; hence, there are many possible confounding factors that cannot be controlled for. Therefore, rather than attempting to control for all variables of interest, the context in which the study is conducted should be described in as much detail as possible. To learn the effects of these variables, we should run replications of case studies. If control over such aspects can prevail through replications of a case study, one may have more confidence in the research outcome. So far there have been relatively few replications of case studies in software engineering, for example Dinh-Trong et al. [2005] conducted a close external replication of the Apache and Mozilla studies in [Mockus et al. 2002] on another open source (FreeBSD) project.

We have investigated the reproducibility of software development projects and products in the context of four commercial companies developing independently an identical medium-sized document management system. We carried out four instances (replications) of a case study in which team size and experience, functional requirements, and customer interactions were controlled for, while the cost and duration of the project varied. This allowed us to gain experience with replicating software engineering case studies.
In the following, we first discuss the types of replication studies in general and case study replications in particular. After that we describe our study in the context of this classification, and comment on practical experiences with replication. We conclude with lessons learned.

2. TYPES OF REPLICATION STUDIES
In addition to the type of study (experiment, case study, survey, etc.), replication studies may be categorized as conceptual, close versus differentiated, literal versus theoretical, and internal versus external.

2.1 Conceptual Replication Study
In order to call a study a replication, at least the research question, including the principal independent and dependent variables, should be repeated from the original study, but the operationalizations of these variables, context variables, measuring instruments, and data collection and analysis methods may be new. This is called a conceptual replication [Shadish & Fuller 1994].

2.2 Close and Differentiated Replications
According to Lindsay and Ehrenberg [1993], close replications attempt to retain, as much as possible, most of the known conditions of the original study. The strict form of close replications is also referred to as exact or direct replications, but they are in practice infeasible in human-intensive studies in software engineering. Even if the same people take part in the replication, they would have had a learning curve since the original study. The major purpose of a close replication is to test the reliability and robustness of the results of the original study.

In contrast, differentiated replications involve variations in essential aspects of the experimental conditions of the study. In studies in software engineering, examples are replacing students with professionals as subjects or use another company if the type of study is a case study. One may also replace the application system being studied, the tasks to be carried out, the support technology used, etc. If the outcome of a study is confirmed in different contexts, one may become more confident that the stated relationships actually hold. In a systematic review of controlled experiments, 18 percent of the identified experiments were replications, of which $\frac{3}{4}$ were classified as differentiated ones [Sjöberg et al. 2005].

2.3 Literal and Theoretical Replications
Yin [2003] distinguishes between literal and theoretical replications. A literal replication is supposed to deliver similar results as the original experiment, while a theoretical replication is supposed to deliver contrasting results but for predictable reasons. In the software engineering literature, replication implicitly means literal replications. The reason is probably that theoretical replications require the testing of theories or comprehensive hypotheses, which is usually not the case in present empirical software engineering [Hannay et al. 2007].

2.4 Internal and External Replications
Internal replications are carried out by the same researchers that conducted the original study. External replications are conducted by researchers that are independent of the researchers and will thus reduce the likelihood of researcher bias and strengthen the overall results. This point may be supported by the findings in the review reported in [Sjöberg et al. 2005]: seven of the eight internal differentiated replications reported results that confirmed the results of the original experiment. For the external differentiated replications, the opposite pattern was found; only one of the seven external replications confirmed the results of the original experiment.

3. REPLICATION OF CASE STUDIES
Analogous to the tradition of replicating an experiment if an interesting finding is uncovered, Yin [2003] states that one should carry out multiple-case studies; that is, for multiple-case studies, there is a replication logic (as opposed to sampling logic), which is described as follows:

Each case must be carefully selected so that it either (a) predicts similar results (a literal replication) or (b) predicts contrasting results but for predictable reasons (a theoretical replication). The ability to conduct, for example, 6 or 10 case studies, arranged effectively within a multiple-case design, is analogous to the ability to conduct 6 to 10 experiments on related topics; a few cases (2 or 3) would be literal replications, whereas a few other cases (4 to 6) might be designed to pursue two different patterns of theoretical replications. If all the cases turn out as predicted, these 6 to 10 cases, in the aggregate, would have provided compelling support for the initial set of propositions. If the cases are in some way contradictory, the initial propositions must be revised and retested with another set of cases. Again, this logic is similar to the way scientists deal with contradictory experimental findings. [Yin 2003]

Ragin [2001] distinguishes between replicated or multiple case studies with respect to timing:

*The key contrast between the study of multiple instances and studying a series of cases is the timing of the selection of the cases. In serial case studies, findings from one case determine the selection of the next case; in the study of multiple instances by contrast, the researcher identifies multiple instances of a phenomenon at the outset of an investigation.* [Ragin 2001]

Although the term “replication” refers to something that is repeated or folded back, the purpose of replicating studies is to perform further investigations into a phenomenon. Whether the multiple studies (being case study, experiment, survey, etc.) are done in series or in parallel, may be less important. An advantage of running them in a series is that a later study can be designed and conducted based on the results and experiences (for example, practical issues) of the previous one. An advantage of running the studies in parallel or independently of each other is that the findings and experiences from one study would not affect the other ones. It is worth noting that case studies are most commonly replicated only in the conceptual form, because it is seldom feasible to find a context that is similar enough to replicate the case study in detail. However, there is not clear boundary between the conceptual and detailed replications, because some of the operationalizations will always be different even in the most detailed replication.

4. A SERIES OF REPLICATED CASE STUDIES
Here we outline our experiences from running a case study four times for which we controlled several important factors. We focus on describing how the case study fits the types of replication and highlight practical considerations in replicating case studies. The detailed report and the reproducibility results are reported in [Anda et al. 2009].
4.1 The Study
The primary research question of the study was to investigate reproducibility of software development (and the lack of it) in a realistic commercial development context by building several identical software systems. We studied reproducibility of the following dimensions of development processes and products: firm price, time schedule, emphasis on analysis and design, contractor-related costs, actual lead time, schedule overrun, reliability, and usability and maintainability.

The scientific reasons for choosing this research objective was to establish the extent of variation in commercial software development that is driven by factors outside contractor control, such as practices employed by different companies, variation among individuals of similar skill, and, most importantly, human creativity expressed in multidue ways the different teams implemented the same system requirements. Furthermore, we wanted to report detailed observations of four commercial projects implementing a system with identical requirements. The methodological reason was to implement the measurement framework of the entire lifecycle of a project from the multifaceted observations that can draw and triangulated from numerous data sources.

The study started with a call for tender sent to 81 Norwegian and international software consultancy companies operating in Norway. Thirty-five of the companies provided bids. Four of the companies were selected to develop individual systems. The selection was done primarily to ensure the successful outcome (projects completing) and to contrast different project and product outcomes based on observed differences provided in the tenders.

To control for team size and developer skills, each of the teams was requested to have two developers of similar professional experience.

To control for interactions with customer, an issue-tracking tool was used for interacting with all the companies in order to attempt to provide as similar feedback as possible to each one.

To enable multiplicity of the analysis, detailed and frequent qualitative and quantitative data was collected and the companies were compensated for the extra effort it took to collect the data to ensure quality.

We found low reproducibility for firm price, time schedule, schedule overrun and reliability and high reproducibility for actual lead time and usability. The main value of the study is the reported observations about each project provided in [Anda et al. 2009] that enable better understanding of some of the factors that cause variation in software development. Figure 1 shows expected values for project and product dimensions together with observed values.

The expectations were driven by variations in the bids. Figure 1 shows that the most “surprises” came from Companies A and B. Company A exceeded all product quality expectations, but had significant time overruns. Company B, on the other hand, satisfied only the reliability expectations, missing all others. Companies C and A mostly performed the role of the least and most expensive contractors, with Company D having relatively high contractor related costs (effort and time needed to interact with the company and accept the finished product) and Company C exceeding low expectations in product usability.

4.2 Categorization of the Study
Our study is a hybrid of differentiated replication, where we varied firm price and the company process, and literal replication, where we tried to keep the product requirements, user interactions, team size and experience as similar as possible. We conducted all four studies ourselves; thus, the replications were internal. Even though we conducted internal replications, we avoided the researcher bias of attempting to confirm the outcome of the original study by running all four studies in parallel; that is, we did not know the outcome of any of the studies before we conducted the other ones.

In a case study, the selection of cases should be driven by the research questions. A vital part of replicating a case study is the choice of additional cases. Given the context of this study, where the research organization was also the software customer, we had a unique control over the choice of case, and consequently this example illustrates well how the cases where chosen and conducted so that several important aspects were controlled to be similar while other where chosen to vary. Three of the factors that we controlled were set in the bids: specification of the requirements, the programming language (Java), and the minimum size of the company. We chose to control the size of company because we were concerned that a very small company would not be able to provide a sufficiently large development team.

The developer skills were controlled to be similar in the process of negotiating contracts with the four companies. We also conducted a separate day-long experiment that confirmed similarity of their performance. Furthermore the customer-contractor interaction was kept identical using the tool BugZero for all communication.

The factors representing differentiated reproducibility included the amount of resources devoted to the project and development processes. The replications were chosen to be dissimilar according to these aspects in order to assess whether differences among the providers (or more simply, the firm price, which was related to the amount of resources companies devoted to the project) would lead to corresponding differences in the projects and products.

5. Lessons Learned
A major challenge related to replicating case studies is to get the opportunity for replication. The ease of replication depends to a large extent on the topic studied. Studies that involve actions of individual programmers on small programs that developers are given to write, are easier to replicate than studies of entire projects. In replications, we will select cases in which past events are similar (or dissimilar if the hypothesis relates to observing of impact of some predictor). For example the study of BSD [Dinh-Trong et al. 2005] is a literal replication of an earlier study of Apache and Mozilla [Mockus et al. 2002]. It was a close external replication where the context was similar, another open source project FreeBSD, and the hypotheses and measures remained the same. The replicated study itself was a differentiated replication comparing project participation, developer productivity, and defect density and resolution time among Apache, Mozilla, and similar-sized commercial projects. Here the contrast was observed between open source and commercial projects. At a finer granularity, a comparison of activities of the same developer with and without a specific editor (Version Editor) was carried out by Atkins et al. [2000]. In some cases such selection is relatively easy, either because of a large choice of projects or because it is possible to impose more control than pure observation allows, which was the case in the study described in this paper.

A related grand challenge is to convince other groups of researchers to carry out a replication. As is also the case for most in-depth empirical research in any discipline, it may be expensive
to run the replication. Attracting sufficient funding, internally or externally to the researcher organization, may be difficult. One possibility is for other researchers, who also teach, to reuse the material for the case study to run it as a student project. The systems produced by the students could then be compared with the ones developed by the various companies. Obviously, this study can be replicated using another requirements specifications (in a differentiated replication) since other research groups may not need this exact system.

Different dimensions and aspects of a software development process may be measured in many ways. Consequently, a challenge in replicated case studies is related to measuring the same dependent and independent variable in different contexts, and each case may lack some of the data sources or the data may be less reliable. For example, while most projects may have a version control system such as CVS, their effort data (time-sheets) are likely to differ in the granularity of tasks for which effort is recorded and in the quality of the data (how accurately the effort is estimated or assigned to a particular task). We have exemplified how this can be solved in part by imposing and paying for a certain amount of control over the projects, as well as by combining and triangulating different data sources acknowledging that available and reliable data sources and measures will vary among projects. In particular, we paid for interviewing the projects participants, for detailed time sheets that recorded information at the same granularity for all four projects, and for access to all produced documents and complete version control records.

The results of our case study demonstrate low reproducibility for many important aspects of software engineering. This shows the need for more case studies and for replications of case studies in software engineering in order to determine aspects that can be replicated, to define concepts and measures of these reproducible aspects, to focus technology and process research on these areas, and, more generally, to obtain fundamental relationships among software inputs, context, and outputs in which we can have more confidence. It also means that many more replications may be necessary.

REFERENCES
Figure 1. Expectations vs. observations