Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System

Bente C.D. Anda, Dag I.K. Sjøberg, *Member*, *IEEE*, and Audris Mockus, *Member*, *IEEE*

Abstract—The scientific study of a phenomenon requires it to be reproducible. Mature engineering industries are recognized by projects and products that are, to some extent, reproducible. Yet, reproducibility in software engineering (SE) has not been investigated thoroughly, despite the fact that lack of reproducibility has both practical and scientific consequences. We report a longitudinal multiple-case study of variations and reproducibility in software development, from bidding to deployment, on the basis of the same requirement specification. In a call for tender to 81 companies, 35 responded. Four of them developed the system independently. The *firm price*, *planned schedule*, and *planned development process*, had, respectively, "low," "low," and "medium" reproducibilities. The contractor's *costs*, *actual lead time*, and *schedule overrun* of the projects had, respectively, "medium," "high," and "low" reproducibilities. The quality dimensions of the delivered products, *reliability*, *usability*, and *maintainability* had, respectively, "low," "high," and "low" reproducibilities. Moreover, variability for predictable reasons is also included in the notion of reproducibility. We found that the observed outcome of the four development projects matched our expectations, which were formulated partially on the basis of SE folklore. Nevertheless, achieving more reproducibility in SE remains a great challenge for SE research, education, and industry.

Index Terms—Software engineering life cycle, software quality, software project success, software process, multiple-case study.

1 Introduction

Suppose that you have developed a well-defined requirement specification for a new software system and that you would like to contract a software development company to build it. An important issue is how much variability there might be regarding the firm price (as opposed to a price based on the number of hours actually spent), planned time schedule, and development process in the bids from different contractors in the current software industry. Ideally, we would like to know to what extent we can expect similar (or better) outcomes if we were to choose software developers with similar (or better) practices and (more) resources. It is a practical problem for the software industry if it is not clear what can be gained by hiring more expensive software contractors. In this paper, this practical problem is restated more formally as a problem of variability and reproducibility.

From a scientific point of view, an understanding of what it means for a phenomenon to be reproducible is important if software engineering (SE) phenomena are to be investigated using scientific method and, hence, if SE is to become a mature industry. The properties of nonreproducible phenomena cannot provide a basis for theories in SE [33]. For example, to assess whether a specific software technology yields desired benefits, for example reducing defects, we need to compare the measured reduction (or increase) in defects from using the technology with the natural variability in defects among similar projects. If the observed impact is not larger than the underlying variability, the technology is not likely to be beneficial.

Moreover, many software development methods assume that certain relationships hold and, hence, that some reproducibility is present, among project dimensions in their de facto use of some variables to predict others. Examples are the use of size to predict costs [12], the use of coupling and cohesion measures to predict maintainability [15], and the generally assumed relationship between development process and outcome in software development projects [40].

The issue of reproducibility has, to our knowledge, not previously been studied in the SE community. To investigate variability in, and reproducibility of, factors that are present in software production, we designed a longitudinal study of software development. The study started with a call for tender for a new Web-based information system to track all the empirical studies conducted by the SE Department, Simula Research Laboratory. The tender was sent to 81 software consultancy companies in Norway, including all large companies. Thirty-five of the companies provided bids. Then, four of the companies were selected to develop the system and to be subject to an in-depth study. We observed large variations in the firm price, planned time schedule, and proposed development processes of the 35 bids. This led us to select four companies that were different with respect to these factors. At the time of

- *B.C.D. Anda is with the Department of Informatics, University of Oslo, PO Box 1080 Blindern, N0-316, Oslo, Norway. E-mail: bentea@ifi.uio.no.*
- *D.I.K. Sjøberg is with the Simula Research Laboratory, PO Box. 134, N0-1325, Lysaker, Norway. E-mail: dagsj@simula.no.*
- *A. Mockus is with Avaya Labs Research, Department of Software Technology Research, 233 Mt. Airy Rd., Basking Ridge, NJ 07920. E-mail: audris@avaya.com.*

Manuscript received 15 Feb. 2008; revised 16 Oct. 2008; accepted 7 Nov. 2008; published online 24 Nov. 2008.
Recommended for acceptance by K. Inoue.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2008-02-0071.
Digital Object Identifier no. 10.1109/TSE.2008.89.

writing, the four systems had been in use by internal and external users for two years.

Because the reproducibility of software production had not been explored previously, we did not have the necessary basis to formulate elaborate hypotheses and predictions and, consequently, the study is exploratory in nature. Nevertheless, if software production is reproducible, it should be possible to observe systematic effects of variations in the inputs to software production. Consequently, we studied how required resources and planned development processes differ given the same requirement specification and how these differences may, in turn, affect how software projects are carried out and the products that are developed.

The study involved four companies with independent professional teams who develop software. We controlled the requirements, team size, developer skill, and the technology environment to ensure that they were as similar as possible across the four projects. We also controlled the interaction between the contractor and developers to ensure that it was as similar as possible for all of the projects.

The four development projects were assessed according to costs, lead time, and schedule overrun. The quality of the resulting systems was assessed according to reliability, usability, and maintainability. The usability assessment was outsourced to a research institute that specializes in human-computer interaction. The maintainability assessment involved hired experts on commercial software development. These dimensions of the projects and products were considered to be the most important from the point of view of a software contractor. We studied the variations in bids and in these project and product dimensions to obtain a measure of reproducibility.

Previous SE case studies have tended not to have either the degree of control over the project and product requirements that we imposed or the ability to follow projects through their entire life cycle that we had in our study. Even though the case study reported in this paper is exploratory and it is not possible to identify causal inferences, it is possible to make a number of observations of scientific and practical importance.

The *methodological* contributions include the measurement framework of the entire life cycle of a project and multifaceted observations that were drawn and triangulated from numerous data sources. The *scientific* contributions involve a better understanding of reproducibility in software production that should provide the basis for SE methods, tools, and studies. The *practical* contributions involve descriptions of detailed scenarios of what can be expected in a software-contracting context and how to use software bids to anticipate project outcomes.

The remainder of this paper is organized as follows: We start from related work in Section 2. The concept of reproducibility is explicated in Section 3. Section 4 introduces the terminology and concepts that were used in this study. Section 5 describes the research methods that were used to design and organize the study, including how the collection of data was validated. Section 6 reports the results of the investigation of the reproducibility of the bids, projects, and products. Section 7 discusses the results for each of the four companies in light of what could be expected on the basis of the information given at the time of bidding. Section 8 discusses the validity of the results of this study. Section 9 discusses the results and identifies methodological, scientific, and practical contributions. Section 10 concludes and outlines directions for further work.

2 Related Work

We have been unable to find other studies that focus on the variability and reproducibility of the outcome of complete software development projects that were carried out by professional developers. The most closely related study examined nine companies who developed the same system, in order to investigate the effect of three development platforms (Java EE, PHP, and Perl) [74]. The systems were evaluated with respect to usability, correctness, robustness, performance, maintainability, and size and structure of the code. The informal way in which these attributes were measured, the different purpose of the study, and a much smaller scale (a maximum of 30 hours in the atmosphere of a noncommercial competition) make it difficult to make any direct comparisons with our study.

In N-version programming (NVP), several individuals or teams develop independently multiple functionally equivalent versions of the same system in order to build fault-tolerant software: "the independence of programming efforts will greatly reduce the probability of identical software faults occurring in two or more versions of the program" [6], [7]. The various versions are executed in parallel and the output given by the majority of the versions is used. In an NVP experiment on the independence of defects [57], [14], 27 students built their own version of a small program. The number of input cases for which more than one program failed was substantially larger than could be expected if the defects in code that caused the program failures were independent. In several cases, the students made the same logical errors when coding. This suggests that failures, at least in student-produced programs, are to some extent reproducible. Unfortunately, the study does not present sufficient detail to quantify the extent to which defects are reproducible.

At Sheffield University, students formed multiple small teams that built systems for commercial clients [41]. For each client, several teams competed to build the system that the client judged to be the best. However, little information was reported on the actual quality of the resultant products and how quality was measured. A more remotely related work compares seven programming languages according to runtimes, sizes, reliability, implementation effort, and program structure, using as a basis 80 implementations of a simple program [73].

The reproducibility of the effect of using specific SE technologies may be manifested in replicated experiments. However, among the 5,453 scientific papers in the SE literature that were identified in a systematic review of SE experiments, only 20 replicated experiments were found [88]. Only one replicated experiment involved subjects that actually built anything (use cases) [23]. Most (seven) replications were conducted in the area of inspection. We have found no replicated case studies on software development in the literature.

The study reported herein investigated how differences in resources and development process affect software projects and products. Related to this is the study of cost models for software development projects. COCOMO is an

example of a cost model and takes as input the size of the project, various product attributes, personnel attributes (reflected by team experience and familiarity with the application), and technology used in the project [12], [13]. Another type of cost model is analogy-based estimation models, which use the cost of completed similar projects as input [84], [92]. Many studies have also been conducted on identifying cost drivers of, and productivity factors in, software development. Examples are [10], [63], [65], [75]. Among the large range of factors that have been identified are the ones that we chose to control in our study: requirements, team size, developer skills, programming language, and contractor-developer interaction.

The developers' skills may be an important predictor of project cost and effort. For example, a study with programs written by volunteer Master's students showed variation of an order of magnitude in the programming time between the fastest and the slowest programmer for a variety of programming languages [71]. Great variability in performance on debugging tasks by programmers who had different experience was reported in [25]. Others have suggested that programmers' experience may not affect their skill to any great extent [29]. A discussion of *when* people learn from experience in general, and in the context of software maintenance in particular, can be found in [53].

Furthermore, maturity of development process has been shown to have a positive effect on project and product outcomes; in particular, lead time, effort, and quality [19], [36], [37], [40], [59], although the benefits may be reduced by the costs incurred by development processes [21]. Studying the impact of the processes' maturity lies beyond the scope of this paper. Assessments of the maturity of development processes are very rare in Norwegian consulting companies, and none of the companies that we studied had been subject to such an assessment.

3 Reproducibility

Scientific inquiry and mature engineering industries are recognized by *reproducibility*. That phenomena and the results of investigating them should be reproducible is a central principle of the scientific method. When evaluating a study, scientists must ask themselves "how reproducible are the findings" [30].

3.1 Definitions of Reproducibility

The precise definition of *reproducibility* varies across disciplines, but it is often closely related to the definitions of *repeatability* and *replicability*. In a recent encyclopedia of philosophy of science [89], 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. For example, in a compendium of chemical terminology [47], reproducibility is defined as "the closeness of agreement between independent results obtained with the same method on identical test material but under different conditions, different operators, different apparatus, different laboratories, and/or after different intervals of time." Furthermore, ISO 5725-1:1994 defines conditions of reproducibility as conditions under which test results are obtained with the same method on identical test items in different laboratories with different operators using different equipment [46].

Although most literature relates reproducibility and repeatability, Hunter [43] introduces a special use of the term "repeatability" to distinguish between those two concepts: "Repeatability is a measure of the variability (imprecision) of a single response within a single laboratory. Reproducibility measures the variability (interlaboratory bias) between measurements of the same response across different laboratories."

In the social and behavioral sciences (as well as in the natural sciences), *reproducibility* is often used as a synonym for *replicability* [5]. For example, Judd et al. [49] state that "replication means that other researchers in other settings with different samples attempt to reproduce the research as closely as possible"; see also [24], [58], [85]; Cohen [22, p. 155] refers to the *reproducibility* of observation statements as a requirement for using them to evaluate universal knowledge claims, but, instead of *replicability*, he uses the term "*reliability*" to denote "the stability of a set of observations generated by an indicator under a fixed set of conditions, regardless of who collects the observations or of when or where they are collected."

When testing for reproducibility, the relevant conditions must be kept fixed; "it does not matter if irrelevant conditions vary" [5]. Wagner [91] refers to the relevant conditions as the *theoretical* ones and stresses that this does not mean that the *empirical* conditions need to be the same. Hence, reproducibility is related to the repeatability of theoretical conditions. Of course, determining which conditions are the theoretical ones may not always be straightforward. The typical SE situation is that an *actor* applies various *technologies* to perform certain *activities* on an existing or planned *software system* [87]. Hence, the theoretical conditions will often relate to these dimensions.

If a phenomenon cannot be reproduced, knowing its properties is unlikely to be very useful, and hypotheses that predict the phenomenon cannot be falsified. An example of how the lack of reproducibility can hinder progress is the setback that a promising new technology of gene microarrays experienced when it was shown to yield widely different results among different microarray platforms [62].

3.2 Reproducibility in Software Engineering

If SE projects and products are not reproducible, then results from one project may not be applicable in any other project. Without reproducibility in the SE industry, software practitioners and contractors would have little control over their projects. Models for assessing processes, such as CMM [68] and ISO/IEC 15504 [81] (formerly SPICE), have been developed as a way of making the SE industry more mature. A study of CMM level 5 projects indicates that "some of the biggest rewards from high levels of process maturity come from the reduction in variance of software development outcomes that were caused by factors other than software size" [1]. Note, however, that very few software companies use CMM [64] and that, of these, very few get the top score [20]. The extent to which such assessments measure actual performance has also been questioned [86]. Moreover, models of process assessment may indicate the level of maturity of a particular organization but do not reflect the practices of the entire industry.

SE projects or products that are not trivially small or simple are unlikely to be reproduced completely in all respects. Every software project has its unique requirements and is carried out by a unique development team in the sense that, even if the same team were to conduct several identical projects, their experience would increase from one project to the next. Consequently, we cannot usually study the reproducibility of complete software projects and products. In this paper, we consider a project or product to have been reproduced if, for a given purpose, it is sufficiently similar to the original project or product. Our overall research question is, therefore, to investigate *the extent* to which key dimensions of software projects and products are reproduced.

3.3 Measuring Reproducibility

Software projects and products are complicated entities that cannot be represented by a single scalar; the study of reproducibility requires a conceptual model and measures of several software dimensions to determine whether the conditions referred to by a concept have been reproduced. We had to answer the following questions:

1. What dimensions of software projects and products should be used to measure reproducibility?
2. What is the best way to measure them?
3. What values indicate low, medium, and high reproducibilities?

The dimensions that one should select to describe software production will, of course, depend on the focus of interest. In the study reported herein, we selected a number of commonly used dimensions that are represented by measures that take only positive values. This choice was made to simplify our assessment of variability. The detailed measures are described in Section 4.

We considered a number of concepts and measures that are related to reproducibility and that are used in several domains. We chose to use the inverse of the coefficient of variation (CV) to quantify the extent of reproducibility in software production for each dimension that characterizes the software project and product:

$$\text{Coefficient of variation}: \quad \text{CV} = \frac{\text{stddev}}{\text{mean}},$$

$$\text{Reproducibility}: \quad \frac{1}{\text{CV}}.$$

The standard deviation represents the variation in (or the lack of) reproducibility. The mean is used to normalize the measure to make comparisons among dimensions that have different scales and measurement units. To measure reproducibility, we invert the CV. This choice is motivated by the intuitive appeal of the idea that a phenomenon that can be reproduced more easily would exhibit lower variability. Note that the CV (also known as an effect size measure in which the change is expressed in standard deviation units) is commonly used in other domains, such as the biological and medical sciences [55]. In biology, the CV is often used for assays (procedures that measure certain properties of biological components). The results of repeated trials of such assays tend to be reported in terms of the CV because the standard deviations of assays generally increase (or decrease) in proportion to the increase (or decrease) in the mean [77].

Meta-analysis is a method for summarizing multiple quantitative research studies and aims to reduce the uncertainty by combining information from individual studies [38]. Analogous to our use of 1/CV to adjust for different scales and units among *multiple dimensions of software production* is the use of, for example, Cohen's d and Hedges' g as measures of effect size in meta-analysis to adjust for possible variations in scale among *multiple studies* [31], [54], [80].

The CV is also used when studying reliability. In this case, it can be used to summarize the distribution of times between failures. The most tractable is the exponential distribution when the mean and standard deviation are equal, making the CV close to 1. Moreover, the CV is most suitable for measures that take positive values.

The only use of the CV in SE that we identified as directly relevant to our study is reported in [28], in which the variability of effort among development phases was measured using the CV.

We would have liked to have used specific universal values to judge the extent of reproducibility from the 1/CV values observed in this and other studies. Unfortunately, such values would depend strongly on the domain and the purpose of the analysis. For example, safety-critical products may require much higher levels of reproducibility in factors that pertain to reliability than products that do not pose any danger to health or safety. Instead, we think it makes more sense to observe values of reproducibility for multiple projects and make a judgement that is based on the particular context. We discuss this further in Section 6.

In addition to measuring reproducibility in absolute terms for various dimensions of software production, we also investigated reproducibility by comparing actual outcomes with our expectations using pattern matching as described in Section 7. If the existing knowledge of the effect of input and context variables on project and product dimensions can be represented in a model, investigating the accuracy of the model would be similar to measuring reproducibility; reproducibility has been achieved if we have variability for predictable reasons. For example, a large variation in the firm price of a bid would not indicate a lack of reproducibility as long as it was possible to accurately predict that the higher the bid, the better the service or product that the customers receive, and the lower the bid, the worse the service or product. The current state of the SE field does not allow the construction of such models, but as the SE field matures, it may be possible in the future.

Errors in measurement may present a challenge when attempting to measure reproducibility; even if processes involved in software production were perfectly reproducible, we would expect to see some variation even for the measures that were controlled to be the same. For example, the reproducibility of medical measurements (see [11]) involves comparison between the error produced by the measurement device and the variance observed in repeated measures of the same property. In our study, we observed some measures, including costs, size, and lead time without error. We controlled a number of project properties and used data validation and triangulation extensively to reduce possible errors for the remaining measures.

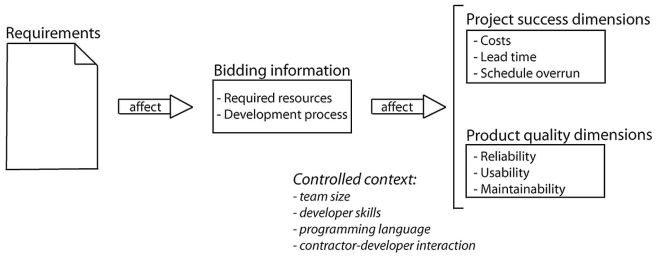

Fig. 1. Relationships between bids and project and product outcomes.

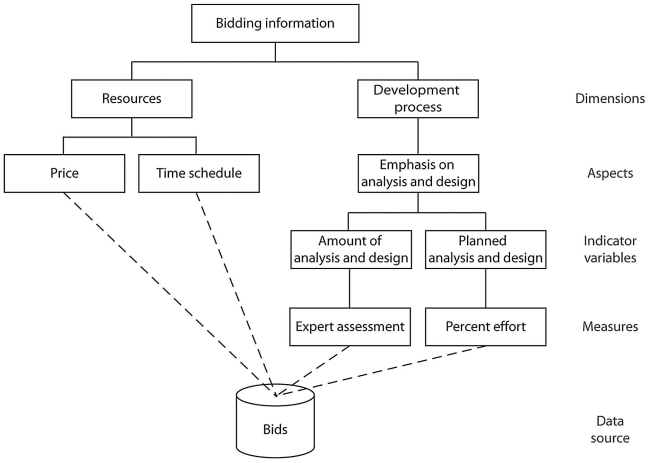

Fig. 2. Information from the bids.

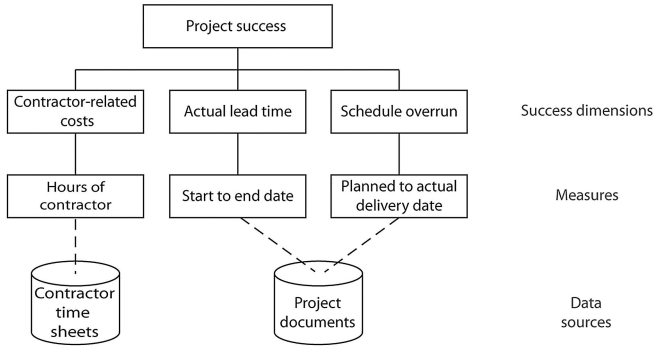

Fig. 3. Project success dimensions.

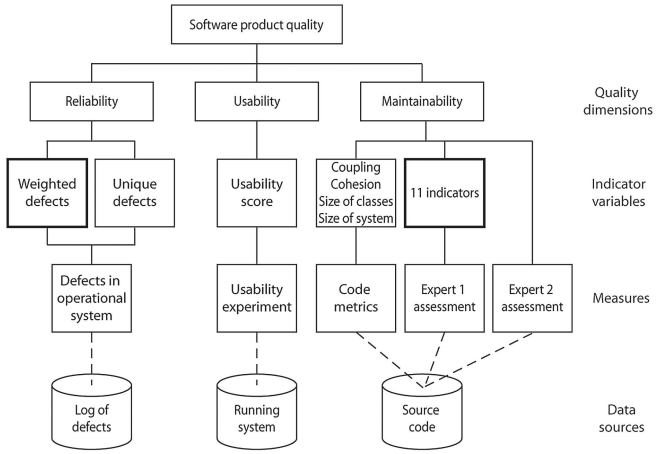

Fig. 4. Dimensions of software quality.

4 Conceptual Model and Measures

The complex nature of SE means that many dimensions of software development cannot be measured directly. Therefore, a challenge in SE studies is to find one or more measures that faithfully represent a given dimension. It is often difficult or impossible to identify a set of measures that completely *represent* a phenomenon. Therefore, measures that only *indicate* or represent some aspects of the phenomenon, called *indicators*, are used [22]. Here, we use the term "measure" for measurements that provide values for the dimensions or for their indicators. The process of defining concepts (or constructs) in terms of observable variables or measures is called operationalization. *Construct validity* represents the extent to which inferences can be made about theoretical constructs on the basis of these operationalizations [83]. Fig. 1 shows the relationships between bids and outcomes that we investigate in this paper. Figs. 2, 3, and 4 show how the dimensions depicted in Fig. 1 were operationalized in our study and how the data sources were used. The data sources are described in Table 11 in the Appendix.

For all companies, the team included two developers and one project manager. The skills of the developers were represented by formal education in programming of at least three years and by three years of industrial experience with the technology used in the projects (individual qualifications are known to have a large impact in SE [16], [26]). Contractors and developers interacted using a bug tracking system and e-mail. All the projects used Java, Javascript, Java server pages with Tomcat, and Mysql as a back end. The projects employed similar development tools.

Other factors could have been controlled for. For example, controlling for lead time would have enabled a direct comparison of how projects deal with time constraints. However, in this study, we could not control for this factor because we needed complete products to evaluate their maintainability.

4.1 Information from the Bids

Bidding for contracts is an important part of the software development business and is typically one of the first steps taken in a software development project. From the point of view of the development organization, the content of the bid is important for winning contracts, attracting business, and achieving business goals. Once a contract is won, the content of the bid lays the foundation for the cooperation with the contractor. From the point of view of the software contractor, it is, of course, important not only to choose the best bid but also to learn whether or not the bid is likely to be fulfilled.

An essential dimension of a bid and, consequently, of an initial plan for a development project is the *resources* required from the developer's point of view. Two principal aspects of resources are *price* and *time schedule* [32]. Measures for these two aspects were derived directly from the bids. Note that the bid price may be affected by business factors in addition to technical factors. For example, different companies may have different business strategies regarding the profit that should be derived from a project; some companies may be willing to bid low to enter a new market. The complete range of business factors that affect a bid is not known and, consequently, attempting to control for them was not feasible and would also have removed some realism from the study. We decided to include the bid

price because it will necessarily have an impact on the way projects are planned and executed, independent of how the companies come up with the bid price.

Different development organizations may employ a range of development processes. A software contractor may also have specific preferences with respect to the software development process that is to be followed in a specific project. It is generally assumed that a relationship exists between the development process and project and product outcomes. A primary objective of SE is to propose development processes that will reduce costs and lead time and improve the quality of software products. However, because of the creative nature of software development, the extent to which differences between success and failure in a project can be attributed to the software production process remains an open question. Consequently, we chose *development process* as the second dimension characterizing the bids.

Definitions of *development processes* are typically comprehensive and include factors that range from project organization and management to the detailed use of particular techniques. For the purpose of our study, we chose to focus on the extent to which analysis and design was emphasized in the description of the development process given in the bids. This aspect was chosen because it was well described in the bids, as opposed to other aspects of development processes, such as quality assurance and testing, which were less often well described. However, in the bids that included detailed descriptions of the planned development processes, the emphasis on analysis and design in the majority of the cases corresponded with the emphasis on other noncoding activities, including quality assurance and testing. Therefore, large amounts of planned effort on analysis and design in most cases implied more planned effort on testing and quality assurance. Furthermore, the extent to which analysis and design should be emphasized in the development process if companies' development practices are to be characterized adequately is a typical challenge in SE.

Given that we could not derive directly from the bids the extent to which analysis and design would be emphasized in a project, we triangulated using two indicator variables. Triangulation is a method for deriving a measure from several lower level measures. It is used widely for designs in which multiple sources of data are used to investigate a question from multiple perspectives [79]. The main objectives of triangulation are to improve construct validity in particular and overall quality in general [94].

Many of the bids contained some information about the *analysis and design* of the product, which had already been carried out at the time of bidding. To measure this variable, we assessed subjectively the amount of analysis and design. Most bids also contained an estimate of the effort to be spent on the project activities. From this information, we were able to calculate the percentage of the total effort that was planned to be spent on analysis and design (*planned analysis and design*) in the remainder of the project.

4.2 Project Success Criteria

A successful project delivers functional and nonfunctional requirements on time and within budget. In our study, we divided the criteria for the success of a project into easy-to-measure dimensions (Fig. 3). *Contractor-related costs* are defined as the effort spent by the contractor. The complete cost of the project from the point of view of the contractor includes both the price specified in the contract and additional costs for the contractor.

Actual lead time is defined as the time from the start of the project until the system becomes operational. Many software products have to be introduced within a restricted time window in order to satisfy business needs.

Schedule overrun is the delay in the delivery of the software product relative to the time schedule specified in the contract. Software projects are notorious for delivering products late and exceeding estimates [48], [67]. A project that is not delivered on time may incur extra costs or disrupt business for the contractor and may have large economic consequences. Note that we did not measure cost overruns because we only investigated projects for which a firm price was given.

4.3 Software Product Quality

According to the ISO/IEC standard 9126 (www.cse.dcu.ie/essiscope/sm2/9126ref.html), a software system has six major dimensions that pertain to quality: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability*, and *portability*. In our study, we compared the systems according to *reliability*, *usability*, and *maintainability*.

Reliability is important because defects in an operational system may lead to such undesirable outcomes as system crashes or corruption of data. We used *weighted defects* as our primary indicator variable (using bold type in Fig. 4) because it best matches our understanding of the concept. The weights indicate the severity of the defects. An additional indicator variable is the number of *unique defects* for each company. This indicator was used to validate the primary indicator variable.

Usability is important because a system that is not perceived as usable may prevent users from achieving their objectives, may discourage future use, or may reduce users' productivity. The result of measuring the usability of a system will depend on the population of target users. We assumed that the target users were unfamiliar with the system and had a limited knowledge of usability issues. We chose this target population because users of the system who are external to Simula will not be familiar with it, while even internal users are not likely to use the system frequently and may, therefore, forget how to perform tasks between the sessions. Experienced users, or users with detailed knowledge of usability, may experience other usability problems. The exact consequences of low usability would depend on the context. For example, if the use of a system is compulsory, its low usability may not affect the extent of use but may reduce productivity. The chosen usability measure relies partly on expert opinion (see Section 6.3.2) but is used in the human-computer interaction community to provide results on an interval scale [34].

Maintainability is important because most systems undergo changes and the costs of maintaining a system often exceed the original development costs. The results of measuring the maintainability of a software system depend on the perspective adopted when measuring. In our study, we adopted the perspective of long-term maintenance. Such maintenance may include large maintenance tasks that are performed by software developers who are experienced with Java but are not thoroughly familiar with the system. We chose this perspective because we no longer have access

to the original developers and we expect the costs of long-term maintenance to dominate the costs of short-term maintenance and, consequently, to have a greater economic impact. The importance of maintainability depends on the type and context of the system. For example, maintainability may be of less concern if the expected life span of the system is short and only small changes, such as error corrections, are anticipated.

We did not consider the remaining three dimensions of ISO/IEC 9126, because the system was tested thoroughly to ensure that the required functionality had been implemented (consequently, only minor differences in functionality would be expected). Efficiency was not important, because the systems were relatively small, had few concurrent users, and managed small amounts of data. Finally, we did not consider portability to be an important issue, because such Java systems can run on most major platforms.

5 Research Method

This section describes the nature of the study, the selection of the companies that participated, the organization of the research project, and the collection of data.

5.1 Type of Study

The unique combination of realism, scale, control, and detail that was achieved in our study warrants some discussion. The unit of study was the software company. The study was conducted in two parts. In the first part, we collected bids from 35 companies; in the second part, we selected four companies for an in-depth study. The study's focus on a set of cases that had a clear identity makes it a case study. According to Yin [93], a case study should be used "when a "how" or "why" question is being asked about a contemporary set of events" that are investigated within its real-life context, and he states that "the case study inquiry

- copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result,
- relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result, and
- benefits from the prior development of theoretical propositions to guide data collection and analysis."

The typical situation of case studies, with more variables of interest than data points, means that it is not possible to control for all possible confounding factors. Instead, the context in which the study is conducted should be described in as much detail as possible.

Even though we had no propositions from tested theories, we followed a *top-down approach* [76] when designing our case study. This approach was used to select projects that maximized the variations in firm price and in proposed process descriptions. This study is also a *multiple-instance* case study (with each development project representing an instance) according to the definition given in [76].

According to Yin, there is one "original" study and the studies that follow (replications) are supposed to deliver either similar (literal) or contrasting (theoretical) results, where the case "predicts contrasting results but for predictable reasons." In our situation, there was no "original" study; we conducted all studies in parallel.

5.2 Selection of Projects

Three of the factors that we controlled were set in the call for tender sent to 81 Norwegian and international software consultancy companies operating in Norway: the specification of the requirements, the Java programming language, and the minimum size of the company. We were concerned that a very small company would not be able to provide a sufficiently large development team.

Thirty-five of the companies provided bids. None of the companies had previous relations with the contractor. In this case, we had a full list of prices and a description of development processes of the 35 companies.

Four of the companies were selected to develop individual systems. This choice of four companies represented a trade-off between having a sufficient number of projects to investigate the effects of differences in required resources and planned development processes and having sufficient means to hire the companies and observe their projects.

In a case study, the selection of cases should be driven by the research questions. To investigate the extent of reproducibility in the second phase, we considered four approaches:

1. Reproducibility is indicated if there exists a set of software providers that produce similar project and product outcomes.
2. Reproducibility is indicated if a random set of providers produce similar results.
3. Reproducibility is indicated if dissimilar providers produce similar results.
4. Reproducibility is indicated if different types of providers produce different results that can be predicted from the differences among the providers. In this context, the differences in the bids predict differences in the projects and systems produced. That is, if we understood underlying cause-effect relationships, we would be able to reproduce different effects if we chose to.

We chose a combination of approaches 1 and 4; the companies were similar with respect to the controlled context and, at the same time, dissimilar with respect to the amount of resources devoted to the project and development processes (see Fig. 1). As an additional criterion, to reduce the risks for our research project, we selected companies that appeared likely to complete the project satisfactorily.

5.3 Practical Organization and Logistics

The SE Department at Simula represented researchers and contractors in the same project. To separate the concerns of researchers and contractors, one team played the role of the contractor and consisted of a project manager and one user representative. Both were employed by Simula and had, respectively, 10 and 5 years of experience in software development in industry but no research experience. The other team was responsible for research and consisted of two researchers and one research assistant. In addition, an experienced consultant worked on an hourly basis to ensure that Simula behaved realistically in the role of a contractor and that the development projects were affected by the research project as little as possible. Three people who were not involved directly in the rest of the project had developed the requirement specification, which focused on

TABLE 1
Overall Reproducibility

	Dimension	No. of companies	CV	1/CV	Reproducibility
Bids	Firm price	35	0.65	1.5	Low
	Time schedule	14	0.49	2.0	Low
	Emphasis on A&D*	27	0.20	4.9	Medium
Projects	Contractor-related costs	4	0.29	3.4	Medium
	Actual lead time	4	0.14	7.1	High
	Schedule overrun	4	0.87	1.1	Low
Products	Reliability	4	0.48	2.1	Low
	Usability	4	0.17	5.9	High
	Maintainability	4	0.46	2.2	Low

*Analysis and design

functionality, but also indicated that the user interface should be similar to that of the general Web system used at Simula. The requirements were perceived by all the companies to be well specified, given the relatively small system. The interested reader may receive a copy of the requirement specification document (11 pages) upon request.

The usability experiment also involved researchers from another institution, SINTEF. Two external senior practitioners provided expert evaluation of the code.

In the contract meetings with the four chosen companies, the companies were, for the first time, told about the research project and that three other companies (the identity of which was not revealed) would develop the same system. Our plans for ensuring privacy and collecting data were also presented. We also obtained the curriculum vitae of each of the proposed team members and after, some negotiation, we chose teams in which all members had approximately three years of experience in software development (cf. skill factor, Fig. 1).

We decided to run the four development projects in parallel, to ensure that Simula as a contractor would behave as similarly as possible with each team and to avoid learning effects in the role of a contractor. One method for helping to ensure that the "contractor" team behaved consistently toward the different companies (cf. factor *contractor-developer interaction*, Fig. 1) was to use an issue-tracking tool, Bugzero, for communications.

5.4 Data Collection

To help ensure that the values of the variables were measured correctly, we made several attempts to control the process by which data was collected. The companies were given extra payment as compensation for the effort needed for research purposes, such as participating in interviews and preparing and sending snapshots of code. Because the contractor was a Simula team, we could easily measure the contractor effort and collect information about the number and kind of defects that occurred after the systems became operational.

6 Results

This section describes the results of our study of the reproducibility found in the bids, projects, and products. Table 1 shows the reproducibility of all measured dimensions.

We used 1/CV as an indicator measure of reproducibility, as explained in Section 3.3. We mapped values of 1/CV to three categories of reproducibility: 1/CV falling in the intervals $(0, 2.5)$, $[2.5, 7)$, and $[7, \infty)$ indicates, respectively, "low," "medium," and "high" reproducibilities. Because of the lack of previous work in this area in SE, we chose this mapping on the basis of the context of the study, observed values of reproducibility measures, and common sense. Other levels may be more appropriate in other domains; for example, drug manufacturing may require greater reproducibility than software production. The particular choices of the three reproducibility levels were made primarily to streamline the discussion.

6.1 The Bids

Table 12 in the Appendix shows the information from the bids for all 35 companies. A firm price was given in all of the bids. The functionality was specified well in the requirements, but none of the companies used a method for functional sizing when bidding. A time schedule was given by some of the companies, but many were probably reluctant to suggest a time schedule at the bidding because this entails additional risk for the company [32]. Most of the bids included an analysis and design of the system, a description of the development process to be used, and an estimate of effort in hours for each project activity.

It was not feasible to study the reproducibility of the description of analysis and design that was included in the bids directly. There were, of course, similarities among many of the diagrams because the functionality was specified well in the requirements, but there were large variations in the level of detail of analysis and design, ranging from nothing at all to one or more of the use cases, screens, architectural description, data models, and navigation diagrams, as well as discussions of nonfunctional requirements.

Many of the companies described their process as incremental. However, there was no time for several increments because of the short duration of the project. In fact, the companies did not develop this system incrementally. Therefore, we disregarded whether the processes were described as incremental or waterfall. Moreover, some of the bids referred to the companies' formal process, but our focus was on their planned process, that is, what they actually planned to do in the project.

To measure the emphasis on analysis and design, we obtained two indicators and combined them to obtain an overall measure; see Fig. 2. To obtain values of the first

TABLE 2
Tender Prices for Six Projects of the Norwegian Public Roads Administration

Construction type	N companies	Stddev	Mean	CV	1/CV
Bridge	2	15,000	45,000	0.3	3.0
Electrical installation	6	13,000	49,000	0.3	3.7
Electrical installation	7	15,000	57,000	0.3	3.9
Road maintenance	5	4,400	47,000	0.1	10.5
Road maintenance	3	2,900	16,000	0.2	5.6
Road maintenance	3	9,800	47,000	0.2	4.8
Mean	**4**	**10,000**	**44,000**	**0.2**	**5.2**

indicator, BidAD ("A&D in bids" in Table 12), one of the authors assessed the amount of analysis and design that was present in each bid using an (ordinal) scale with the following values: 0 = "None," 1 = "Very brief," 2 = "Brief," 3 = "Detailed," and 4 = "Very detailed." To obtain values of the second indicator, PlanAD ("Planned effort on A&D" in Table 12), we used the estimates (in hours) of effort planned for different project activities (provided in most of the bids) to calculate the percentage (relative to total effort) of the *planned* effort for analysis and design in the project.

We judged both components to have similar effects on the overall measure of analysis and design. Therefore, to equalize the scales for these two measures, the PlanAD measures were divided by 10 before being added to the BidAD measure, which resulted in a range of values from 0.0 to 5.0 that was similar to the range of values for BidAD. In our opinion, the slightly larger range of values (and correspondingly larger weight) for planned effort is justified by the fact that the projects that had a detailed analysis and design in their bids still planned to expend some effort on these activities during the project (from 5 to 11 percent of the total effort). The resulting values (BidAD + PlanAD/10) are shown in the column "Emphasis on A&D" in Table 12.

6.1.1 Reproducibility of Bids

We found no relationship between the *firm price* and *planned time schedule* for the companies that provided *lead times*. Therefore, the *lead time* appears to be affected mostly by the work style and by the amount of resources that the company said that they would dedicate to the project. Correspondingly, there was no relationship between the *firm price* and the process measures. This is probably because a substantial part of the analysis and design was completed as a part of a bid and, therefore, did not directly influence the price of the project. Yet, it may be surprising that the general focus on process and analysis and design does not appear to affect the *firm price*. Basic statistics, including the CV and 1/CV, are shown in Table 1. The overall measure of analysis and design is much more reproducible than its components (BidAD and PlanAD). However, the projects that provided planned effort (only these projects are included in the overall measure) may be slightly more homogeneous: If we were to restrict our sample to these 27 projects, the 1/CV measure for BidAD would increase from 2.3 to 2.5. The bulk of the increase can be explained by the fact that BidAD is correlated negatively with PlanAD (-0.72 Pearson correlation and -0.69 Spearman correlation), and as a result, their sum has less variance than each component. The negative correlation reflects the simple observation that if more of the analysis and design is done in the bid, less of the analysis and design remains to be done during the project.

The bids made for the study, including the large variations in firm price, were also described in [52]. The large variations in firm price were confirmed in another study of software development projects, in which 30 companies from 11 countries in Eastern Europe and Asia presented their bids [50]. The 1/CV of the price was 1.6.

Using raw data that was obtained from a study that compared programming languages as a basis [72], [73], we were able to calculate the reproducibility of the development effort in hours for 80 implementations of a small system with different programming languages. The overall 1/CV was 0.87, while the language-specific 1/CV ranged from 1.1 for Java to 2.1 for the C language.

The projects in our study were small, but small software projects are common. To get a better understanding of the reproducibility of software projects, we compared the projects in our study to projects within the same price range from a collection of 279 projects of the Norwegian Public Roads Administration in the period 1996-2006 [70]. Table 2 shows that the average 1/CV of these road construction projects was more than three times higher than those found in the two SE studies described above (1.5 and 1.6, respectively). The higher reproducibility measure that was found in road construction projects may be due to the relative maturity of the civil engineering discipline.

We investigated the values for reproducibility relative to the size of the project. The average 1/CV for all the 279 road construction projects was 10.8 (average price of 3.9 million euros); that is, a 10-fold increase in size yields approximately a doubled value for 1/CV. For the six largest projects (average price of 71 million euros), the average 1/CV was 16.7; that is, a 1,600 times increase of project size yields a three times increase in the value for 1/CV. A similar pattern may exist for software projects as well.

6.1.2 Selected Companies

Four companies were selected to implement the system, using the method described in Section 5.2. We defined the exclusion criteria for selection as follows:

- Companies that were too small to provide a team of three members were excluded. To reflect typical development projects, we wanted teams to develop the systems, rather than individual developers. Note

TABLE 3
Characteristics of the Companies

	Company A	Company B	Company C	Company D
Nationality	Norwegian	Norwegian	Norwegian	International
Ownership	Private	By employees	By employees	Listed on exchanges
Location	Oslo	Oslo	Bergen	Oslo + 20 countries
Size	Appr. 100	Appr. 25	Appr. 8	Appr. 13,000 worldwide
Firm price	€20,000	€45,380	€8,750	€56,000
Agreed time schedule	55 days	73 days	41 days	62 days
Planned effort on A&D	28%	20%	7%	23%

that apart from requiring that three people should be involved in the project, we did not place any constraints on how the teams cooperated and on whether the team members worked full time or part time.

- Companies that gave no, or a brief, or very brief, description of the analysis and design of their proposed solution were excluded (see Table 12). The companies were asked to sketch a solution in their bids. Instead of attempting to maximize the variation in the selected companies with respect to emphasis on analysis and design, we decided simply to ensure a certain level of description of analysis and design, because we assumed that to be a reasonable indicator that the companies would deliver a satisfactory system.
- Companies that proposed solutions that were based on the use of existing systems for publishing information on the Web were excluded. We wanted to study software development projects that created the entire source code from scratch, yet based on a Web technology platform that was in widespread use at the time (Autumn 2003).
- Companies that did not provide a time schedule or sufficient information to calculate the effort that they planned to expend on analysis and design were excluded (see Table 12).

After these criteria had been applied, eight companies remained. We decided to select four of them by maximizing the variation in price. The companies were, respectively, 5, 13, 30, and 34 in Table 12. It turned out that there was also a substantial spread among these companies with respect to time schedule and planned effort on analysis and design.

Table 3 summarizes the four companies with respect to nationality, ownership, location, and size at the time of the study, as well as the firm price, agreed time schedule, and the extent to which planned development process was emphasized. All this information was provided in the bids.

Contract meetings were held with all the four companies. For the most expensive company, the firm price was renegotiated to a slightly lower value. The Simula contractor team considered, on the basis of their experience with software bidding, that the bid included a margin for bargaining and that, in order to behave as a realistic contractor, it was necessary to negotiate that price. Many of the companies included the company's experience with relevant technology and similar applications in their bids. Our selected companies had sufficient experience, but we also asked to see the curricula vitae of the potential developers to verify that they had the necessary skills. We required a minimum of three years of education in computer science and a minimum of three years of experience in industrial software development. For one of the companies, we had to negotiate to get sufficiently qualified developers. For all of the companies, detailed plans were made for the start and delivery dates. Table 3 refers to the price and plans agreed in the contract meetings. These differ somewhat from the price and planned time schedule in the bids. The agreed time schedule was the basis for calculating the overrun of the projects.

All four companies provided a detailed analysis and design in their bids, but the effort that they planned to expend on these process activities varied. Therefore, only the planned effort is given below. The companies were asked to follow their planned process within reason. They were monitored closely during the development, both by Simula's contractor team (who received regular status reports) and by the researchers (snapshots of all documents were sent from the companies to the researchers on a weekly basis, and the research team had weekly interviews with the developers). Consequently, we were able to ensure that the actual emphasis on analysis and design corresponded to the planned emphasis.

6.2 The Project

Table 4 shows the overall outcomes from the four development projects. Sections 6.2.1 and 6.2.2 provide justifications for the scores of the three project dimensions, while Sections 6.3.1, 6.3.2, and 6.3.3 describe the three product dimensions.

6.2.1 Contractor-Related Costs

Contractor-related costs were measured in terms of the numbers of hours spent by the contractor on each of the projects (see Fig. 3). Table 5 shows how the effort was expended. The contractor's project manager spent time on fulfilling the contracts, clarifying requirements, and testing and installing the systems. The contractor's user representative spent time on clarifying requirements and testing and took part in design meetings. Technical support on the contractor side was required during delivery and installation. The difference in the number of hours spent by the different companies was validated by considering the amount of e-mail exchange between the Simula project manager and the development teams and by considering the total number of issues recorded for each company.

TABLE 4
Quality of Project and Product

Dimensions		Company A	Company B	Company C	Company D
Project	**Contractor-related costs**	90 hours	108 hours	155 hours	85 hours
	Actual lead time	87 days	90 days	79 days	65 days
	Schedule overrun	58%	23%	93%	5%
Product	**Reliability**	Good	Good	Poor	Fair
	Usability	Good	Fair	Fair	Good
	Maintainability	Good	Poor	Poor	Good

TABLE 5
Contractor Effort in Hours

Contractor effort	Company A	Company B	Company C	Company D
Project manager	42	61	73	41
User representative	37	6	68	30
Technical support	11	21	14	14
Total	**90**	**108**	**155**	**85**

TABLE 6
Classification of Defects

Severity Level	Definition	Weight	Examples in this study
Severity 1	Critical situation/ System Down	8	None
Severity 2	Severe Impact	4	Defects that may lead to system crashes
Severity 3	Moderate impact	2	Corrupted data (those observed here were not serious)
Severity 4	Minimal impact	1	Defects in displayed text, some of which may affect usability

6.2.2 Actual Lead Time and Schedule Overrun

The actual lead time was measured in calendar days from the start to the end date of the project. The overrun was measured by the number of calendar days that the actual lead time exceeded the planned time schedule, and in percentages: ((actual lead time − planned time schedule) ∗ 100)/planned time schedule}. Table 4 shows the actual lead times and overruns.

6.3 The Product

6.3.1 Reliability

To measure software reliability, we used the number of defects that were detected after the systems became operational. We applied four levels of severity, as indicated in the Orthogonal Defect Classification [18], [44], see Table 6. To simplify the assessment of reproducibility, we aggregated the number of defects in the various levels of severity into a single scalar by giving weights to each level. We used expert assessment to agree on the multiplier of 2 from one level of severity to the next for this project, which is in accordance with, for example, the practice of the SEI CMM Level 5 company Syntel. Clearly, projects with different priorities may need different weightings. For example, projects that are concerned exclusively with the cost of fixing defects may choose equal weighting (assuming that each defect incurs, on average, the same fix costs). In projects that are mostly concerned with the cost of downtime, most of the weight may be placed on defects that may cause outages.

Fig. 5 shows that Systems A and B had few defects at severity levels 2 and 3; System C had many defects at these two levels; and System D had many defects at level 3. The overall reliability of the systems, calculated in terms of the weighted defects, was in the range of 10 to 31. A 1/CV of 2.1 indicates low overall reproducibility. This is confirmed by considering how many defects were detected in more than one project. Only one common defect was reported for the four systems; a defect of severity level 3 that was reported for both Systems C and D. We conclude that System C had poor reliability, System D had fair reliability, and Systems A and B had good reliability.

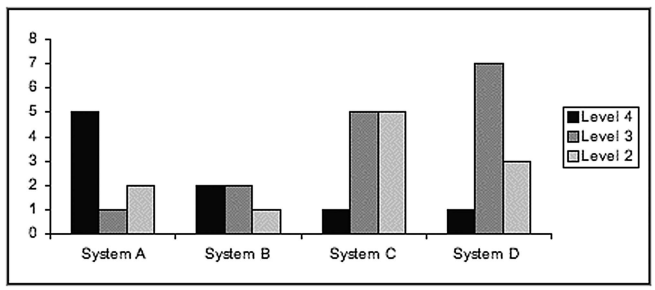

Fig. 5. Defects detected in operational use.

TABLE 7
Summary Statistics for the Systems (Mean and Standard Deviation Shown for WMC1, OMMIC, and TCC)

	System A	System B	System C	System D
Coupling (OMMIC)	7.7, 15.8	5.3, 11.8	8.6, 25.0	4.7, 14.1
Cohesion (TCC)	0.26, 0.37	0.17, 0.31	0.20, 0.23	0.11, 0.22
Size of classes (WMC1)	6.9, 11.2	7.8, 10.3	11.4, 12.5	4.9, 4.5
Size (LOC)	7937	14549	7208	8293
Size (NOC)	63	162	24	96

6.3.2 Usability

The usability of the four systems was evaluated in an experiment. Details of the experimental design, variables, and measures can be found in [34]. Three core user tasks were identified for use in the experiment, using the requirement specifications for the requirements and acceptance test as a basis. The tasks included predefined inputs and covered tasks for the internal users (who may add information) and for the external users (who seek information). These tasks were chosen because they were judged to be the most challenging from the perspective of usability. The participants in the experiment were representative of the system's user population (17 PhD students and one scientific programmer working at Simula). Thirteen had no previous knowledge of the system, while five had superficial knowledge without having tried any of the four versions of the system beforehand.

Two compound usability measures, task usability and global usability, were calculated as follows:

$$\begin{aligned} Task\ usability &= Task\ completion - User\ error\ rate \\ &\quad - Task\ time + Satisfaction \\ Global\ usability &= (Task\ usability\ 1 + Task\ usability\ 2 \\ &\quad + Task\ usability\ 3)/3. \end{aligned}$$

The global usability scores were, respectively, 113, 86, 84, and 117 for Systems A to D.[1] Higher scores indicate greater usability. These scores were constructed from expert opinions (assessed using a five-point Likert scale) that were added over different dimensions, such as the time needed for a user to complete a task and the number of defects. Even though the global usability score may have a negative value in theory, a constant can always be added so that our formula can be used for the reproducibility measure.

The CV for the global usability score resulted in a value of 0.17, which indicates high reproducibility. However, it is worth noting that, although the differences in global usability scores indicate overall reproducibility, there were differences among the systems for the different tasks. For example, although System D received a high global usability score, it received poor task usability scores on Task 2. Correspondingly, System C received a lower global usability score because of a very low score on Task 1, although the System C team did very well on Tasks 2 and 3. We conclude that the usability of the systems from Companies B and C was fair, while the usability of the systems from Companies A and D was good.

1. Note that, in the report [34], Company B is called Company C and vice versa.

6.3.3 Maintainability

The IEEE standard for SE terminology defines software maintainability as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or to adapt to a changed environment" [45]. The operationalization of this definition remains a challenge. To the authors' knowledge, there are no empirically validated methods for assessing the maintainability of complete software systems that are generally applicable and no studies that compare the maintainability of different systems. However, the maintainability of (parts of) individual systems has been studied extensively. Maintainability may be affected by a large number of factors in addition to properties of the code, such as the qualifications of the maintainers, the maintenance tasks, and the tools used. In this case, we wanted to assess maintainability mainly on the basis of the code.

Three principal strategies for measuring maintainability are 1) the calculation of structural measures based on static analysis, 2) expert assessments, and 3) the use of benchmarks, although none of these strategies are well established or have been evaluated empirically in a multisystem context.

In the benchmark strategy, a set of "representative" changes is defined and performed on the different systems [4]. The effort required to implement the changes is then an indicator of the maintainability of each of the system. We are not aware of any reports of the application of such a benchmark to evaluate maintainability, but a "maintenance benchmark" was used to evaluate various development tools [51]. For the purpose of this study, we decided to assess maintainability using a combination of structural measures and expert assessments. A justification for this approach can be found in [2], [9].

Assessment based on structural properties. Several sets of structural measures have been proposed to assess code maintainability. The most common ones are the CK metrics [17]. It has been determined that this set can be used to predict the maintainability of individual classes in object-oriented systems [27]. We decided to use an adapted version of a subset of the CK metrics that we found most suitable for our study. Table 7 shows the mean and standard deviation for the metrics WMC1 (number of methods per class, each method has a weight of 1), OMMIC (call to methods in unrelated class), and tight class cohesion (TCC), as well as lines of code (LOC) and number of classes (NOC) for the four systems. The depth of inheritance is not included, because Company C did not use inheritance at all and the other systems mostly used only one level of inheritance.

TABLE 8
Expert's Characteristics and Assessments

Characteristic	System A	System B	System C	System D
Choice of classes	1	0	1	1
Design adapted to system	1	0.5	0	1
Three-layer architecture	0.5	0.5	0	0.5
Use of components	0.5	0.5	0	1
Encapsulation	0.5	1	0	1
Inheritance	0.5	0	0	0.5
Simplicity	0.5	0	0.5	0.5
Naming	0.5	0.5	0	0.5
Comments	1	0.5	0.5	1
Good use of class libraries	1	0	1	0
Appropriate technical platform	1	0	0	0
Total score	**8**	**3.5**	**3**	**7**

Table 7 shows that the systems were substantially different in terms of LOC, numbers of classes and the distribution of functionality over the classes (as can be seen from the large differences in the values for WMC1). Furthermore, the values for OMMIC and TCC show that the developers of the systems chose different trade-offs with respect to a focus on good coupling versus good cohesion.

There are few empirical studies on the effects of these trade-offs in object-oriented design. Therefore, it is difficult to combine the metrics in Table 7 into one overall measure of the maintainability of a complete system. If we assume that the system with the best mean and standard deviation for the class-level measures is also the most maintainable, we arrive at the following ranking: System D appears to have good maintainability; Systems A and B have fair maintainability; while System C is assessed to have poor maintainability due to high values and large standard deviations for the size of classes and coupling of classes.

However, if we take the size of the systems into account, it may be reasonable to assume that System B will be more difficult to maintain than is indicated by its class level values, due to its large size, while System C may be easier to maintain.

The Maintainability Index (MI) has been proposed for assessing the maintainability of complete systems [69]. The MI uses a polynomial to combine the average per class of four traditional code measures (lines of code, number of comments, cyclomatic complexity, and Halstead Volume) into a single-valued indicator of maintainability. Although the use of the MI has been reported for several projects, there are no indications in the literature that this formula is generally applicable. Consequently, we decided to use the measures shown in Table 7, which had greater empirical support, and to combine them in a more flexible way.

Expert assessment. The experts chose their own criteria for evaluation, using as a basis their experience in software development. The first expert had 25 years of experience, including 10 years with Java development. He was hired for 60 hours to inspect the code of the four systems. The second expert had 10 years of experience, including six years with Java. He was hired for 16 hours. The high costs of hiring these experts prevented us from having two equally thorough evaluations. Due to the simplicity of the four systems, the experts were also asked to attempt to foresee the consequences of design decisions for larger maintenance tasks and for longer term maintenance. The first expert considered the 11 characteristics listed in Table 8 to be indicators of the maintainability of these systems.

The second expert gave each system a score and provided a brief motivation for the score given. The motivations contained some of the same characteristics as those that had been identified by the first expert. Although the two experts did not communicate in any way, their conclusions were similar. Consequently, the opinions of the second expert were used to supplement the detailed insights of the first.

Table 8 shows the first expert's assessment of the four systems. For each of the characteristics, the experts commented on whether it was handled in a satisfactory manner (score 1), to some extent satisfactory (score 0.5), or unsatisfactory (score 0). The overall measure for maintainability was obtained by adding the values for the different scores, with all the characteristics being given equal weight.

Both experts assessed the maintainability of Systems A and D to be good. System A was ranked to be the best. However, the experts commented that the developers of Company D had implemented a much larger system because they, presumably, had started the project with high ambitions of building a system that would be easy to maintain. These ambitions were not fulfilled, which resulted in a low score for some of the characteristics. Nevertheless, for large changes, System D is likely to be more maintainable than System A.

The maintainability of Systems B and C was assessed as low in each case. Again, the experts commented that it was likely that the degree of maintainability would depend on the types of changes that were required. The design of System B was too complex and comprehensive for the application that was ordered. It may have been more appropriate for a larger product. The developers of Company C had not emphasized good design, but, because their system was small, it may be easy to perform small maintenance tasks on it. However, it may be difficult to make larger extensions.

Overall assessment. The CV for the total score in Table 8 is 0.46, which indicates low reproducibility of maintainability. We supplement the results in Table 8 with those in Table 7. This combination yields an overall assessment of maintainability of the systems as follows (summarized in Table 4):

- We consider System D to have good maintainability, judging from both the expert opinions and the structural measures.
- We consider System A to have good maintainability, judging from the expert opinions. The structural measures indicate that System A is slightly less maintainable than System D if the size of the systems is not considered. However, we assume that the relatively small size of System A would give it an advantage in the maintenance phase.
- We consider System B to have poor maintainability, judging from the expert opinions. The structural measures of coupling and cohesion are relatively good, which indicates a maintainable system. However, the system is relatively large and we assume that size will have a negative impact on maintainability.
- We consider System C to have poor maintainability, judging from both the expert opinions and the structural measures.

7 Expectations versus Observations

As shown in the previous section, we found that several of the project and product dimensions had low reproducibility overall. However, we deliberately selected four companies to develop the system that exhibited large variations with respect to required resources and planned development process. In order to have reproducibility, these variations should lead to some predictable and tangible differences in the projects or products (see Section 3.3). Nevertheless, we are not aware of any established model that uses resources and planned development process to predict project and product outcomes. Existing prediction models in SE typically have a very fine granularity. For example, models have been proposed for predicting the reliability of software architectures on the basis of structural design information [78] and for predicting maintenance performance on the basis of specific object-oriented metrics [8]. However, such prediction models require much more detailed information about the systems than is usually available in bids, which means that such models are unsuitable for making predictions at the time that bids are evaluated.

Instead, we used *pattern matching*, where theoretical propositions are compared with the outcome of a study. This technique is commonly used in case studies in the social sciences. In our study, we applied a *visual approach* to pattern matching [90]. Ideally, the propositions should be based on well-founded theories. However, given that there are no SE theories relevant to the topics of this paper [35], our propositions (or the term that we find more suitable here, *expectations*) are based on our interpretation of SE folklore (common and often unstated assumptions).

7.1 Expectations

We started with the common-sense assumption that having sufficient resources for a project should have a positive impact on project and product outcomes. The concept of "sufficient resources" includes an agreed schedule with sufficient time for contingencies and a sufficiently high firm price, which may allow for dedicated full-time project participants. Furthermore, the quality of a software product should be influenced (significantly) by the development process. After all, a substantial part of SE is dedicated to the design of software development processes that (presumably) lead to better quality. An emphasis on development processes should also have an impact on other project and product outcomes, for example, through better control over the project, which, in turn, should reduce the need for contractor involvement and lead to lower overruns.

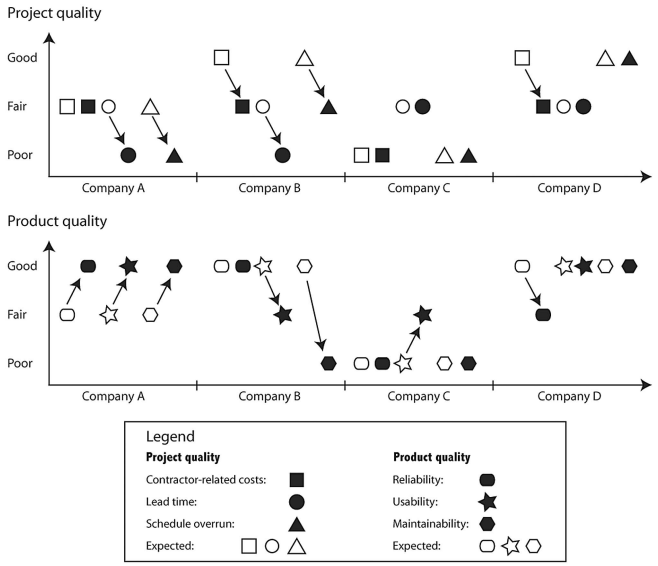

Fig. 6. Expectations versus observations.

Using this assumption as a basis, the project dimensions *contractor-related costs* and *overrun* were likely to be *high*, that is, *poor*, for Project C (shown in Fig. 6 as an unfilled square box and circle, respectively), because the project had few resources and little emphasis on development process. The same two dimensions were likely to be *low*, that is, *good*, for Projects B and D, because they had the most resources and a greater focus on development process. Company A had relatively few resources but placed great emphasis on the development process. Consequently, the contractor-related costs and overrun were likely to be *medium*, that is, *fair*.

All companies agreed on a time schedule at the start of the project (shown in Table 3) and, given the small size of the system and the carefully controlled context, it was reasonable to expect relatively small differences in the actual *lead time*. (Indeed, the actual lead time was found to have high reproducibility; see Table 1.) Moreover, given that overruns on the time schedules were expected for the companies with the lowest lead-time estimates and that overrun seems intrinsic to most software projects, we expected the lead-time dimension to be fair for all of the companies.

The outcome values for the three product dimensions were likely to be *good* for both Systems B and D because both companies had ample resources for this project. They also placed a similar emphasis on development processes. The product outcomes of System A were likely to be *fair* because, though their strong emphasis on development process should improve product quality, their more limited

resources may not be sufficient to take advantage of the better process. The product outcomes for System C were likely to be *poor*, because of low resources and the lack of emphasis on the development process.

7.2 Observations

This section discusses the observations for the project and product quality relative to the expectations given above. We first describe how we arrived at the scores for the various dimensions and then we describe the actual performance of the companies.

7.2.1 Scores

The observations for the *product* quality dimensions of reliability, usability, and maintainability are shown in Fig. 6 as, respectively, filled rounded rectangles, stars, and hexagons. The scores are taken directly from Table 4. On the other hand, the results of the *project* quality dimensions shown in Table 4 were measured in numbers. However, to compare the observations with our expectations, we need to describe the results at a more coarse-grained granularity, because we had no underlying model that could support us in indicating an accurate level of expectations. Therefore, we decided to subjectively use the good-fair-poor scale also for the project quality dimensions.

We arrived at the categories shown in Fig. 6 as follows: Regarding *contractor-related costs*, Company C required 155 h from Simula, which we consider *poor* for this relatively small project. It should have been possible to develop this system without too much intervention from us, so the values of the other companies (85-108 hours) indicate a *fair* performance. The average estimate of the 14 companies that estimated *lead time* (see Table 11) was 54 days. Between 20 and 50 percent more than this value, we find Companies D (65 days) and C (79 days), which we thus find *fair*. Companies A and B have an observed lead time that is over 50 percent more than those 54 days, which we consider to be *poor*. In a study of 42 software development projects, the *mean schedule overrun* was 25 percent [66]. Only two projects had overrun more than 58 percent (Company A, Table 4). Comparing these results with the results of our study, we consider Company D as good, Company B as fair, and Companies A and C as poor on this dimension.

7.2.2 Performance of the Companies

Fig. 6 shows that Company A performed better than expected on the three product dimensions but had a longer lead time and greater schedule overrun than expected. Halfway through the project, the project manager told us that he was tempted to cancel the project to avoid losing more money. One reason why the project was underestimated may be that the consultant who provided the estimate was not involved in any other aspects of the project. Previous studies on the accuracy of software estimates show that if estimators are not responsible for the project, their estimates may be less accurate [61]. Nevertheless, Company A decided to complete the project according to their usual standards to preserve their reputation. Due to the fact that the agreed time to completion was too short, Company A had to change developers in the middle of the project because the developer who was originally assigned to the project left for a long vacation. This may have also contributed to a large overrun. In this particular case, Simula obtained a better product than expected because the company spent much more resources than planned. Nevertheless, if this had been a large project, it would probably have been necessary to renegotiate the contract and, therefore, to pay a higher price to complete the project.

Company B obtained lower scores than expected for most of the dimensions. Investigating this project in more detail did not give us a simple answer to why this happened, but we identified two factors that may have contributed:

- The developers of Company B, as well as of Company C, were located in different offices and, to some extent, worked only part time on the project. (The developers of Company D worked full time on the project and worked in the same room. Company A had one developer working full time on the project, but he was replaced during the project.) According to CVS check-ins, two developers from Company B committed their code at different times and, consequently, may have cooperated very little. Project organization may be an important contributor to project success or failure. For example, full-time focus on one project at a time and colocated team members are encouraged in the agile software development method Scrum [82], and team coordination in distributed teams has been associated with large delays on individual tasks [39].
- Company B derived the method for their analysis and design from a book on SE [60]. We suspect that this choice was motivated by their belief that Simula expected them to use such a method, even though we encouraged them to use the method that they themselves found most suited to the project. Furthermore, the developers may have used this method too rigidly and may not have been sufficiently qualified to use it.

The observed scores for Companies C and D on the quality dimensions matched the expectations reasonably well. Usability for Company C was *fair* in contrast to the expected *poor*. The high overall reproducibility of the global usability score indicates that this dimension is not strongly affected by project resources and an emphasis on analysis and design.

Given the lack of theories in this area, the small sample of projects, and that many aspects of software development are not taken into account in this study, one cannot expect an exact match between expectations and observations. Nevertheless, at least in our opinion, the relatively good fit of Companies C and D, and the fact that the exception for Company A can be easily explained, indicate that the variations to the inputs of software production are, to some extent, mirrored in the outputs. Of course, the desirable goal of constructing a reasonably accurate prediction model that would be useful for decision making in the software industry will require much more work (see Section 10).

8 Validity of Results

This section discusses issues of the validity of this study, divided into construct validity, internal validity, external validity, and repeatability [56], [83], [93].

TABLE 9
Bidding Information from the Two Groups

Dimension	Group	N	Mean	StDev	Minimum	Maximum	CV	1/CV
Firm price	1	17	34725	17182	9940	69060	0.49	2.0
	2	18	20767	15729	2630	69940	0.77	1.3
Planned time schedule	1	7	70.86	19.26	45	94	0.27	3.7
	2	7	36.86	21.56	14	77	0.58	1.7
Emphasis on A&D	1	15	4.67	0.71	4	6.3	0.15	6.6
	2	12	4.55	1.20	2	6.0	0.26	3.8

8.1 Construct Validity

Three issues concerning construct validity are 1) how accurately the values of the variables are measured, 2) how different variables are aggregated to measure the dimensions of software development that were considered in our study, and 3) how well those dimensions represent the constructs that they are supposed to operationalize.

The accuracy of the measurements was strengthened by the implementation of several methods to control the process of data collection. The companies were paid for the extra effort needed to support the research (see Section 5.4). The effort expended by the contractor and the number and type of defects were easily measured, because the contractor was a Simula team that also performed the acceptance tests. Furthermore, the communication with the team was recorded in Bugzero and e-mail.

Due to the exploratory nature of the study and the lack of an established framework for measuring project and product characteristics, there are challenges with many of the measures that we used and with how these are aggregated that may represent threats to the construct validity. Our measures of emphasis on analysis and design, reliability, usability, and maintainability rely, to some extent, on expert assessment. Computing the sum and calculating the mean of values on an ordinal scale obtained through expert assessment may be questioned from a measurement perspective (see, for example, [33]), but is a common practice both in SE and in the social sciences. Furthermore, our measures of emphasis on analysis and design, and usability are the results of aggregating multiple criteria on different scales. We chose to measure these dimensions using aggregations instead of expert evaluation because it provided more transparency on how the measurements were derived and allowed the triangulation and combination of different measures. The usability evaluation was conducted by the HCI group of the research institute SINTEF. The aggregation technique used to derive the usability measure was their standard practice.

The operationalization of important dimensions of software development used in our study, shown in Figs. 2, 3, and 4, are, to some extent, pragmatic and chosen on the basis of what could be measured in the projects. One measure, development process, was particularly difficult to operationalize. As described in Section 4.1, we chose emphasis on analysis and design as described in the bids. We followed the four projects closely during the development phase and found that the actual effort on development process activities, other than coding, to reflect the emphasis on analysis and design in the bids. Nevertheless, it is a threat to the validity of our results that emphasis on analysis and design represents only one, although important, aspect of development processes and that there are challenges involved in the measurement of this dimension, as described above.

8.2 Internal Validity

Threats to internal validity are not a major concern in our study because most of the results are descriptive [93]. However, the discussion of expectations (Section 7) makes some tentative assumptions and offers some explanations for which internal validity *is* a concern.

There is also a challenge to the internal validity of the results on the reproducibility of bidding information. The bidding phase of the study reported herein was preceded by another study on the same bids [52]. Due to the research questions of that study, there were differences in the conditions under which the bids were given. Half of the companies, group 1, had taken part in a prestudy and had also made an earlier bid for the system that was based on a less comprehensive requirement specification, while the other half, group 2, had only made one bid. Table 9 compares the relevant values for the two groups.

Several rounds of bidding are not common for projects of this size. However, the variations are consistently lower for the companies in group 1. Therefore, these results do not, in our opinion, invalidate the results presented in Section 6. Rather, they indicate that the reproducibility of bidding information in practice may be slightly lower than that reported in Table 1.

Companies B and D were in group 1, while Companies A and C were in group 2. Having two companies in each group was desired by the person responsible for the study on bidding reported in [52]. This means that the higher firm price and planned time schedule for Companies B and D could, to some extent, be attributed to the fact that they bid twice. However, we could have chosen four companies from either of the two groups with approximately the same bid characteristics as those of the four companies we actually studied. For example, the most expensive company belonged to group 2 (Table 9). Consequently, we do not consider this difference in how the bids were produced to be an important threat to internal validity.

In Section 7, we showed that the outcomes for project and product quality matched, to some extent, our expectations, which were formulated on the basis of resources and the planned development process. One alternative explanation for the outcomes for project and product quality is the *variation in skills* of the project participants. Therefore, we selected teams with similar qualifications and inspected the curricula vitae of the potential developers to ensure that

they had the necessary skills. Nevertheless, it is still possible that there could be smaller differences in qualifications among the developers in the different companies than we tested for. Such differences may explain some of the differences in the outcome variables.

Development tools may also have an impact, but in our study, the companies used similar tools. For example, all companies used Ant for build management and CVS for configuration management. Three out of four companies used MS Visio for modeling, while the fourth, Company B, used Poseidon. Another study showed no significant effects of tools on project outcomes as long as the developers were familiar with the tool that they used [3]. However, we suspect that the fact that Company B used a more specialized tool for design than the others contributed to their making a more complex design.

Furthermore, we believe that the relatively few observed defects may be a threat to our results regarding reliability.

8.3 External Validity

External validity refers to the issue of "establishing the domain to which a study's findings can be generalized" [56]. Our call for tender was sent to a majority (81) of the software development consultancy companies located in Norway (some small Norwegian companies were omitted to get a good distribution of small, medium-sized, and large companies). Of the 81 companies, 35 responded (a response rate of 43 percent). Consequently, we expect that a large majority of Norwegian software development companies that were interested in conducting this project responded.

This bidding process allowed us to select representative projects in a top-down manner so that all cases had "equal voice and the results [were] not skewed toward the most prominent cases, the cases with the most or best data, or the cases that the researchers simply [happened] to know best" [76]. We observed large variations in the bids and we chose four companies that represented the breadth of price, time schedule, and development process. We had no previous business or personal relationships with anyone in these companies. Furthermore, an external consultant who has many years of experience in industry was employed by Simula to ensure that Simula behaved realistically as a software contractor. He was involved in making the call for tender, in the selection of companies, and in the contract meetings. He also participated in all the project meetings at Simula.

The *requirement specification* specified usability more clearly than it did the nonfunctional requirements reliability and maintainability, as explained in Section 5.3. This might have contributed to higher reproducibility for usability than for reliability and maintainability, although we believe this to be typical for requirement specifications.

The fact that the companies were subjects of research may represent a threat to the external validity of the results, because the projects may have been conducted differently from how projects in the companies were normally conducted. However, the project members did not know the research questions or how the data from the study would be used. Therefore, they could not modify their behavior intentionally in order to manipulate the results. Interviews that were conducted regularly during the project and other contact with the companies indicate that the project teams were concerned primarily with the economic aspects of the project and were not concerned with, or influenced by, being a part of a research study. The only exception was Company B, which we expect would have followed a slightly different process, as explained in Section 7.2.

Generalizing the results of this study and making claims about reproducibility and variability of the development of larger systems is difficult. This is because there may be counteracting effects where on the one hand, larger scale could mean larger variability, while on the other it could mean better opportunities for adjusting the project outcomes, which could lead to larger reproducibility. In the studies on civil engineering referred to above, the larger road construction projects showed much larger reproducibility than did the smaller projects.

One particular challenge with generalizing the results of our study to international software development is the fact that none of the companies had been assessed according to CMMI. CMMI level 2 should ensure repeatability within the company, but CMMI is not common in Norway, as commented in Section 3. Consequently, it was infeasible to hire companies that had been subject to CMMI assessment.

Given the foregoing, we expect that it is possible to generalize the results of our study to Norwegian and international consultancy companies located in Norway within the given scope [87]; that is, small but realistic Java systems developed by small and reasonably qualified teams.

8.4 Repeatability

The repeatability (or reliability) of a study represents the ability of other investigators (and even by the original investigators themselves) to follow the same procedures, to perform exactly the same study in relevant respects, and to arrive at the same findings and conclusions [93]. Yin [93, p. 38] claims that "in the past, case study research procedures have been poorly documented." A systematic review confirms that this is also the case in SE [42]. In our study, all the procedures and documents for collecting data are available from the authors, and the analysis procedures are well documented (Section 6). Therefore, we consider the repeatability of the results of our study to be good.

9 Contributions

This section discusses the results with respect to methodological, scientific, and practical contributions.

9.1 Methodological Contributions

From the methodological perspective, we expect that this work will serve as a basis for further studies. Few case studies in SE have been conducted according Yin's principles [93]. Our study of real SE phenomena combines comprehensiveness and detail with control over important theoretical conditions (see Section 3.1). Furthermore, it is an example of a multiple-case study, a research method that is seldom used in SE.

Our attempts to operationalize fundamental SE concepts to evaluate concrete complete systems revealed deficiencies in the current state of how concepts in the field are understood and employed. The product quality dimension *maintainability* was particularly difficult to operationalize. We proposed a number of concrete dimensions and measures for use in descriptions of software practices and

TABLE 10
Costs Including Contractor's Effort

	Company A	Company B	Company C	Company D
Costs	€25,370	€51,860	€18,020	€61,070

product (Figs. 2, 3, and 4). Partially, the reproducibility of SE projects and products may be improved by using concepts that are directly related to project and product outcomes when describing planned development processes.

The extensive use of diverse data sources exemplifies how to integrate a variety of data and how to ensure the quality of data from quantitative and qualitative sources using triangulation, control, and validation (see Section 4).

9.2 Scientific Contributions

The extent of observed variability was great. Although some of our measures showed high reproducibility and some of the variability in the outcome measures can be explained by the variations in the inputs, our study indicates a lack of reproducibility in SE projects and products, which is in contrast to other domains. For example, in Norway, prices for the construction of roads, for projects of approximately the same size as the software projects that we studied, have higher reproducibility (see Section 6.1.1). This lack of reproducibility should also be taken into account in empirical SE studies. In particular, the lack of reproducibility implies that greater efforts need to be made in studies of software construction to ensure that the contexts of study are well defined so that the theoretical conditions will be stable and hence scientifically sound.

The observed differences in reproducibility means that some outcome variables appear to be better candidate variables for empirical studies of software technologies than others, because higher reproducibility of outcome variables makes it easier to observe whether the technology under study has an effect. Outcome variables with low reproducibility, on the other hand, need either better control to increase reproducibility or further clarification and better operationalization.

The lack of reproducibility in the bidding process suggests that there are important drivers of costs and effort that may not be well described using current SE concepts. Even though bidding is partly a business domain, the bids for software projects contain a substantial SE component. This aspect of the bids has, so far, received little attention in the SE literature. A better conceptualization of the different aspects of the bidding process, for example, including the business aspects, may be necessary to improve reproducibility.

9.3 Practical Contributions

This study has quantified various aspects of software construction and exemplified trade-offs between various project and product dimensions when a software system is bid for and built by an external software development organization. Given the modest size of the systems and the detailed description of the functional requirements, we observed large variations in bids and project and product outcomes.

The functional requirements were described well in the call for tender, but the nonfunctional requirements were described in less detail. The large variations in the design decisions made by the various companies suggests that nonfunctional requirements, especially requirements related to maintainability, should be specified in the call for tender in order to help reduce the uncertainty in project and product outcomes.

We would expect the firm price to capture the differences in requirements and work practices that, in turn, lead to differences in the outcomes. In such situations, high variability would be welcome because it would provide the contractor with a wider choice of desired outcomes, assuming that the developers deliver accordingly better products or services. However, in our study, information about work practices and ambitions regarding project and product quality could only, to a limited extent, be derived from the bids, and the outcomes were not monotonic in price. Our recommendation is therefore that such information is stated more explicitly in the bids and not just be an implicit part of the firm price.

In our study, there were differences in the effort and qualifications required from the contractor. This represents an important additional cost of software projects. In our case, the large variations in the firm price are substantially reduced if the total costs of a project are used instead. Assuming a reasonable internal cost of 60 euros/h, the effort spent by the contractor can be converted into costs shown in Table 10. The total cost for Company D is only 3.4 times more expensive than for Company C, while Company D is 6.4 times more expensive than Company C if only the firm price is taken into account (see Section 6.1.2). This shows that the total costs of procuring a system may be affected strongly by the contractor's effort. Including such costs may alter the differences in price among bidders and may affect the outcome of procurement decisions.

Even after taking into account the contractor's effort, the system from Company C was by far the cheapest. Consequently, this system may represent the best choice for a contractor that is tolerant to delays, that has sufficient qualifications, and that places lower demands on certain quality aspects.

Note also that the case of Company B shows that having sufficient resources is not, in itself, enough to ensure the quality of software development projects and products. The development team must also have sufficient focus on this project and be able to choose the right method for the specific project.

A measure that remained relatively constant among the projects was the actual lead time. This is somewhat surprising, because all the projects were delivered late and the amount of time that the developer was told to spend on our project ranged from full-time work in Company D to only part-time work in the remaining companies. An intuitive expectation regarding actual lead time would, in our opinion, be that if it differed among the companies, it would be shorter for the companies that have a shorter agreed time schedule than for the companies that have a longer agreed time to completion. However, the

TABLE 11
Data Sources

Data source	Description
Access logs	The logs from Simula's web server were collected during the two years the systems were operational.
Bids	The companies' offered firm price was included in all the bids. The time schedule of the project, the planned development process, and the analysis and design of the product were included in many of the bids.
Contractor time sheets	Simula's contractor team recorded the time they spent on the project.
CVS	At the completion of the development and testing, the researcher team received complete CVS bases from the projects.
E-mail	All the e-mail communication between Simula's project manager and the development projects was recorded.
Interviews	The projects' team members were interviewed weekly about their work on the project and about the possible effects of being the object of research. The interviews were semi-structured and based on an interview guide in which some questions were the same each week, while others varied depending on the status of their project.
Issue tracker	The companies registered their questions and needs for clarification in Bugzero; see http://www.websina.com/bugzero/. The Simula contractor team registered their responses. Later on, the Simula team registered defects (classified according to severity) that were detected in the acceptance tests.
Log of defects	This is the log of the defects found after the systems became operational.
Project documents	These are documents related to overall project management, such as time schedules, design descriptions, acceptance test logs, and technical documentation.
Running systems	The four systems.
Snapshots	Snapshots of all documents (including code) were sent to the research team weekly.
Source code	The source code of the four systems.

results show no correlation between the agreed time to completion and actual lead time, and the shortest actual lead time was spent by Company D, which had the second longest agreed time to completion.

10 Conclusions and Future Work

The primary goal of SE research is to study and improve the way software is produced. To achieve this goal, it is crucial to quantify the impact that various aspects of software production have on the success and quality of software projects and products. There are numerous assumptions and a lot of folklore that purport to explain how software production works, but a deep and detailed understanding of the relationships is absent, due to a lack of studies that quantify the effects of software technology under realistic and rapidly changing circumstances. Most studies conducted in SE are relatively small and investigate only isolated aspects of software production in experimental settings that cannot capture the complexity of how software is developed in practice.

We have reported a study of 35 bids and four complete commercial projects that built functionally equivalent software products. To gather detailed information on the software production, we collected and analyzed management information and software development documents and conducted extensive interviews with developers and project managers. Considerable effort was put into the conceptual definition and operationalization of fundamental SE terms. Factors that are known to affect software productivity and quality, including team size, developer skills, development technology, and interaction with the customer were controlled to be similar. Many of the remaining factors varied substantially. We found little reproducibility in the firm price of bids, and in particular, we showed that the variation in firm price was about three times greater than in the more mature domain of road construction. The study shows that these variations in both firm price and variations in schedule and planned development process can, to some extent, be used to predict project and product outcomes, although these outcomes were not monotonic with respect to the resources used. We speculate that the lack of reproducibility of bids, plans, and project and product outcomes may be due partly to the paucity of standards for describing process and product quality.

Given the exploratory nature of our study, the findings should be interpreted as hypotheses to be investigated further. Studies should be conducted that involve more companies and that investigate substantially different contexts of software production, for example, in much larger projects, in different application domains, or in projects that use different development technology or have different organizational structures. Furthermore, reproducibility in

TABLE 12
Bid Information

Company	Firm price without VAT (Euro)	Time schedule (days)	A&D in bids	Planned effort on A&D (%)	Emphasis on A&D
1	2630	14	Brief (2)		
2	4380		Brief (2)		
3	4880		Very brief (1)		
4	4970	28	Brief (2)	30	5.0
5	8750	18	Detailed (3)	7	3.7
6	9940		None (0)	40	4.0
7	11810		Brief (2)	0	2.0
8	11880	94	Detailed (3)	26	5.6
9	12190	77	Very detailed (4)	5	4.5
10	16630		Brief (2)	12	3.2
11	18130		Very brief (1)		
12	18510	91	Brief (2)	20	4.0
13	20000	30	Detailed (3)	28	5.8
14	20020		Very brief (1)	50	6.0
15	21090		Very brief (1)	44	5.4
16	25310		Very detailed (4)	11	5.1
17	33250	49	Detailed (3)	26	5.6
18	25810		Very brief (1)		
19	25940		Brief (2)	20	4.0
20	25980		Very detailed (4)	8	4.8
21	26880	45	Detailed (3)		
22	28700	77	Very detailed (4)	10	5.0
23	28950	42	Brief (2)	30	5.0
24	29000		Brief (2)		
25	33530		Brief (2)		
26	33880	77	Detailed (3)	10	4.0
27	33900		Detailed (3)	11	4.1
28	34500		Very brief (1)	36	4.6
29	38360	63	Detailed (3)	20	5.0
30	45380		Detailed (3)	10	4.0
31	52310		Brief (2)	27	4.7
32	56900		Detailed (3)	14	4.4
33	60750		Brief (2)	43	6.3
34	69060	49	Detailed (3)	23	5.3
35	69940		Detailed (3)	6	3.6

the maintenance phase should be studied, because to do so would complete the picture of the entire life cycle. This is something that we plan to do for the four systems in this study.

Our results suggest that making SE processes, projects, and products more reproducible is a challenge for SE research, education, and industry practice. Future research on reproducibility and fundamental relationships in SE will require that comprehensive comparative studies are conducted with a certain level of scientific rigor. Such studies may require more resources than are available in most SE research groups. At Simula, we have explicitly chosen to use our research funds to support realistic empirical studies, such as the one reported in this paper, rather than to maximize the number of research staff. Moreover, one approach to increasing the research output from a given amount of resources is to outsource the participation in studies to professionals in countries with lower costs. For example, in an estimation study being conducted by Magne Jørgensen at the time of writing this paper, 40 companies from Eastern Europe estimated the effort to develop the system that was the object of the study reported in this paper. Hiring these 40 companies to develop the system will cost approximately the same as that which we paid the four companies in our study (130,000 euros). Another possibility for reducing the costs of such a study would be to use teams of students.

Nevertheless, we believe that investment in adequate empirical studies will ultimately yield economic gains for the software industry. Our vision is that research funding in SE should reflect the importance of software in modern society, which means that the funding should be at a level comparable to that which is found in other disciplines, such as physics and medicine.

APPENDIX

Data sources are summarized in Table 11 and bid information is contained in Table 12.

ACKNOWLEDGMENTS

Conducting such a large case study would not have been possible without contributions from a large number of people. The authors would like to thank M. Jørgensen for his effort made in the bidding phase and for insightful comments on the research; P.E. Arnstad, S. Amundsen, and J. Dzidek for input on evaluating the quality of the code; H.C. Benestad and E. Arisholm for support on the analysis of code structure; C. Mallows for discussion on measuring reproducibility; H.C. Benestad for managing the contractor project; G. Carelius for support on requirements and technical issues; G. Farley for support on requirements and advice during the project; A. Følstad, J. Heim, and J.H. Skjetne for evaluating the usability; S. Grimstad for acting as contractor representative; S.E. Hove for conducting interviews and validating all data received from the development projects; V.B. Kampenes for help with the interviews; B. Manum and T. Gruschke for support regarding data from the Norwegian Public Roads Administration; and the four companies for their effort made in this project. The authors would also like to thank B. Kitchenham and J. Hannay for comments on an earlier version of this paper, the anonymous referees for valuable comments, and C. Wright for proofreading this paper.

REFERENCES

[1] M. Agrawal and K. Chari, "Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects," *IEEE Trans. Software Eng.,* vol. 33, no. 3, pp. 145-156, Mar. 2007.

[2] B. Anda, "Assessing Software System Maintainability Using Structural Measures and Expert Assessments," *Proc. 23rd Int'l Conf. Software Maintenance,* pp. 204-213, 2007.

[3] E. Arisholm and D.I.K. Sjøberg, "Evaluating the Effect of a Delegated versus Centralized Control Style on the Maintainability of Object-Oriented Software," *IEEE Trans. Software Eng.,* vol. 30, no. 8, pp. 521-534, Aug. 2004.

[4] E. Arisholm and D.I.K. Sjøberg, "Towards a Framework for Empirical Assessment of Changeability Decay," *J. Systems and Software,* vol. 53, no. 1, pp. 3-14, 2000.

[5] H. Atmanspacher and R.G. Jahn, "Problems of Reproducibility Complex Mind-Matter Systems," *J. Scientific Exploration,* vol. 17, no. 2, pp. 243-270, 2003.

[6] A.A. Avižienis and L. Chen, "On the Implementation of *N*-Version Programming for Software Fault Tolerance during Execution," *Proc. IEEE Int'l Computer Software and Applications Conf.,* pp. 149-155, Nov. 1977.

[7] A.A. Avižienis, "The Methodology of *N*-Version Programming," *Software Fault Tolerance,* M. Lyu, ed., John Wiley & Sons, 1995.

[8] R.K. Bandi, V.K. Vaishnavi, and D.E. Turk, "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics," *IEEE Trans. Software Eng.,* vol. 29, no. 1, pp. 11-87, Jan. 2003.

[9] H.C. Benestad, B. Anda, and E. Arisholm, "Assessing Software Product Maintainability Based on Class-Level Structural Measures," *Proc. Seventh Int'l Conf. Product-Focused Software Process Improvement,* pp. 94-111, 2006.

[10] J.D. Blackburn, G.D. Scudder, and L.N. Van Wassenhove, "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers," *IEEE Trans. Software Eng.,* vol. 22, no. 12, pp. 875-885, Dec. 1996.

[11] J.M. Bland and D.G. Altman, "Statistical Methods for Assessing Agreement between Two Methods of Clinical Measurement," *The Lancet,* vol. 327, no. 8476, pp. 307-310, 1986.

[12] B.W. Boehm, *Software Engineering Economics.* Prentice Hall, 1981.

[13] B.W. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Eng.,* vol. 1, no. 1, pp. 1-24, 1995.

[14] S.S. Brilliant, J.C. Knight, and N.G. Leveson, "Analysis of Faults in an *N*-Version Software Experiment," *IEEE Trans. Software Eng.,* vol. 16, no. 2, pp. 238-247, Feb. 1990.

[15] L.C. Briand, J. Daly, and J. Wuest, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Trans. Software Eng.,* vol. 25, no. 1, pp. 91-121, Jan./Feb. 1999.

[16] R.E. Brooks, "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology," *Human Aspects of Computing,* vol. 23, no. 4, pp. 207-213, 1980.

[17] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Software Eng.,* vol. 20, no. 6, pp. 476-493, June 1994.

[18] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M.Y. Wong, "Orthogonal Defect Classification—A Concept for In-Process Measurement," *IEEE Trans. Software Eng.,* vol. 18, no. 11, pp. 943-956, Nov. 1992.

[19] B.K. Clark, "Quantifying the Effects of Process Improvement on Effort," *IEEE Software,* vol. 17, no. 6, pp. 65-70, Nov./Dec. 2000.

[20] *CMMI Maturity Profile September 2006,* Software Eng. Inst., 2006.

[21] A. Cockburn, "Selecting a Project's Methodology," *IEEE Software,* vol. 17, no. 4, pp. 64-71, July/Aug. 2000.

[22] B.P. Cohen, *Developing Sociological Knowledge: Theory and Method,* second ed., 1989.

[23] K. Cox and K. Phalp, "Replicating the CREWS Use Case Authoring Guidelines Experiment," *Empirical Software Eng.,* vol. 5, no. 3, pp. 245-267, 2000.

[24] L.J. Cronbach, *Designing Evaluations of Educational and Social Programs.* Jossey-Bass, 1983.

[25] B. Curtis, "Substantiating Programmer Variability," *Proc. IEEE,* vol. 69, no. 7, p. 846, 1981.

[26] B. Curtis, "By the Way, Did Anyone Study Any Real Programmers," *Empirical Studies of Programmers,* E. Soloway and S. Iyengar, eds., pp. 256-262, 1986.

[27] D. Darcy and C.F. Kemerer, "OO Metrics in Practice," *IEEE Software,* vol. 22, no. 6, pp. 17-19, Nov./Dec. 2005.

[28] A. De Lucia, E. Pompella, and S. Stefanucci, "Assessing the Maintenance Processes of a Software Organization: An Empirical Analysis of a Large Industrial Project," *J. Systems and Software,* vol. 65, no. 2, pp. 87-103, 2003.

[29] T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams.* Dorset House, 1987.

[30] H.F. Dingman, "Scientific Method and Reproducibility of Results," *Multivariate Behavioral Research,* vol. 4, no. 4, pp. 517-522, 1969.

[31] T. Dybå, E. Arisholm, D.I.K. Sjøberg, J.E. Hannay, and F. Shull, "Are Two Heads Better than One? On the Effectiveness of Pair Programming," *IEEE Software,* vol. 24, no. 6, pp. 10-13, Nov./Dec. 2007.

[32] E.F. Easton and D.R. Moodi, "Pricing and Lead-Time Decisions for Make-to-Order Forms with Contingent Orders," *European J. Operational Research,* vol. 11, no. 3, pp. 57-67, 1997.

[33] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach,* second ed. PWS, 1998.

[34] A. Følstad and J. Heim, "Usability Evaluation of Four Functional Identical Versions of DES (Database of Empirical Studies)," SINTEF Report SINTEF_A309, 2006.

[35] J.E. Hannay, D.I.K. Sjøberg, and T. Dybå, "A Systematic Review of Theory Use in Software Engineering Experiments," *IEEE Trans. Software Eng.,* vol. 33, no. 2, pp. 87-107, Feb. 2007.

[36] D.E. Harter, M.S. Krishnan, and S.A. Slaughter, "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science,* vol. 46, no. 4, pp. 451-466, 2000.

[37] D.E. Harter and S.A. Slaughter, "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science,* vol. 49, no. 6, pp. 784-800, 2003.

[38] L.V. Hedges and I. Olkin, *Statistical Methods for Meta-Analysis.* Academic Press, 1985.

[39] J.D. Herbsleb and A. Mockus, "Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering," *ACM SIGSOFT Software Eng. Notes,* vol. 28, no. 5, pp. 138-147, 2003.

[40] J.D. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, "Software Quality and the Capability Maturity Model," *Comm. ACM,* vol. 40, no. 6, pp. 30-40, 1997.

[41] M. Holcombe, T. Cowling, and F. Macias, "Towards an Agile Approach to Empirical Software Engineering," *Proc. Workshop Empirical Studies in Software Eng.,* pp. 33-48, 2003.

[42] N.E. Holt, "A Systematic Review of Case Studies in Software Engineering," MSc thesis, Univ. of Oslo, 2006.

[43] J.S. Hunter, "The National System of Scientific Measurement," *Science,* vol. 210, no. 4472, pp. 869-874, 1980.

[44] *IBM Software Support Handbook.* IBM, Version 4.0.1, May 2008.

[45] *IEEE Standard Glossary of Software Engineering Terminology.* IEEE, 1990.

[46] *ISO 5725-l first ed. 1994-I 2-15,* Accuracy (Trueness and Precision) of Measurement Methods and Results, 1994.

[47] *IUPAC Compendium of Chemical Terminology,* second ed., A.D. McNaught and A. Wilkinson, eds. (Royal Soc. Chemistry, Cambridge, U.K.), http://www.iupac.org/publications/compendium/index.html, 1997.

[48] A.M. Jenkins, J.D. Naumann, and J.C. Wetherbe, "Empirical Investigation of Systems Development Practices and Results," *Information and Management,* vol. 7, no. 2, pp. 73-82, 1984.

[49] C.M. Judd, E.R. Smith, and L.H. Kidder, *Research Methods in Social Relations,* sixth ed. Harcourt Brace Jovanovich, 1991.

[50] M. Jørgensen, "The Effects of the Format of Software Project Bidding Processes," *Int'l J. Project Management,* vol. 24, no. 6, pp. 522-528, 2006.

[51] M. Jørgensen, S.S. Bygdås, and T. Lunde, "Efficiency Evaluation of CASE Tools—Methods and Results," TF R 38/95, Telenor FoU, 1995.

[52] M. Jørgensen and G.J. Carelius, "An Empirical Study of Software Project Bidding," *IEEE Trans. Software Eng.,* vol. 30, no. 12, pp. 953-969, Dec. 2004.

[53] M. Jørgensen and D.I.K. Sjøberg, "Impact of Experience on Maintenance Skills," *Software Maintenance: Research and Practice,* vol. 14, no. 2, pp. 123-146, 2002.

[54] V.B. Kampenes, T. Dybå, J.E. Hannay, and D.I.K. Sjøberg, "A Systematic Review of Effect Size in Software Engineering Experiments," *Information and Software Technology,* vol. 49, nos. 11/12, pp. 1073-1086, 2007.

[55] K. Kelley, "Sample Size Planning for the Coefficient of Variation from the Accuracy in Parameter Estimation Approach," *Behavior Research Methods,* vol. 39, pp. 755-766, 2007.

[56] L. Kidder and C.M. Judd, *Research Methods in Social Relations,* fifth ed. Holt, Rinehart, and Winston, 1986.

[57] J.C. Knight and N.G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multiversion Programming," *IEEE Trans. Software Eng.,* vol. 12, no. 1, pp. 96-109, Jan. 1986.

[58] D.R. Krathwohl, *Social and Behavioral Science Research.* Jossey-Bass, 1985.

[59] M.S. Krishnan, C.H. Kriebel, S. Kekre, and T. Mukhopadhyay, "An Empirical Analysis of Productivity and Quality in Software Products," *Management Science,* vol. 46, no. 6, pp. 745-759, 2000.

[60] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process,* second ed. Prentice Hall, 2001.

[61] A.L. Lederer and J. Prasad, "Software Management and Cost Estimating Error," *J. Systems and Software,* vol. 50, no. 19, pp. 33-42, 2000.

[62] E. Marshall, "Getting the Noise Out of Gene Arrays," *Science,* vol. 22, no. 306, pp. 630-631, 2004.

[63] K.D. Maxwell, L. Van Wassenhove, and S. Dutta, "Software Development Productivity of European Space, Military, and Industrial Applications," *IEEE Trans. Software Eng.,* vol. 22, no. 10, pp. 706-718, Oct. 1996.

[64] S. McConnell, "I Know What I Know," *IEEE Software,* vol. 19, no. 3, pp. 5-7, May/June 2002.

[65] K. Milis and R. Mercken, "Success Factors Regarding the Implementation of ICT Investment Projects," *Int'l J. Production Economics,* vol. 80, no. 1, pp. 105-117, 2002.

[66] K.J. Moløkken-Østvold, "Effort and Schedule Estimation of Software Development Projects," PhD thesis, Univ. of Oslo, 2004.

[67] K.J. Moløkken-Østvold and M. Jørgensen, "A Comparison of Software Project Overruns—Flexible versus Sequential Development Models," *IEEE Trans. Software Eng.,* vol. 31, no. 9, pp. 754-766, Sept. 2005.

[68] M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber, "Capability Maturity Model for Software, Version 1.1," Report CMU/SEI-93-TR-24, Software Eng. Inst., Pittsburgh, 1993.

[69] P.W. Oman and J.R. Hagemeister, "Construction and Testing of Polynomials Predicting Software Maintainability," *J. Systems and Software,* vol. 24, no. 3, pp. 251-266, 1994.

[70] H. Pedersen, "Tender Prices: Bridge, Tunnel, Electro and Road Building and Maintenance 1998-2006," Technology Report 2468, Norwegian Public Roads Administration, 2006.

[71] L. Prechelt, "The 28:1 Grant/Sackman Legend Is Misleading, or: How Large Is Interpersonal Variation Really?" Technical Report 1999-18, Universität Karlsruhe, Fakultät für Informatik, 1999.

[72] L. Prechelt, "An Empirical Comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a Search/String-Processing Program," Technical Report 2000-5, Universität Karlsruhe, Fakultät für Informatik, 2000.

[73] L. Prechelt, "An Empirical Comparison of Seven Programming Languages," *Computer,* vol. 33, no. 10, pp. 23-29, Oct. 2000.

[74] L. Prechelt, "Plat_Forms 2007: The Web Development Platform Comparison—Evaluation and Results," Technical Report B-07-10, Freie Universität Berlin, Institut für Informatik, 2007.

[75] J.D. Procaccino, J.M. Verner, K.M. Shelfer, and D. Gefen, "What Do Software Practitioners Really Think about Project Success: An Exploratory Study," *J. Systems and Software,* vol. 78, no. 2, pp. 194-203, 2005.

[76] C.C. Ragin, "Case-Oriented Research," *Encyclopaedia of the Social and Behavioural Sciences,* vol. 3, pp. 1519-1525, 2001.

[77] G.F. Reed, F. Lynn, and B.D. Meade, "Use of Coefficient of Variation in Assessing Variability of Quantitative Assays," *Clinical and Diagnostic Laboratory Immunology,* vol. 9, no. 6, pp. 1235-1239, 2002.

[78] R.H. Reussner, H.W. Schmidt, and I.H. Poernomo, "Reliability Prediction for Component-Based Software Architecture," *The J. Systems and Software,* vol. 66, no. 3, pp. 241-252, 2003.

[79] L. Richards, *Handling Qualitative Data—A Practical Guide.* Sage Publications, 2005.

[80] R. Rosenthal and M.R. DiMatteo, "Meta-Analysis: Recent Development in Quantitative Methods for Literature Reviews," *Ann. Rev. Psychology,* vol. 52, pp. 59-82, 2001.

[81] T.P. Rout, K. El Emam, M. Fusani, D. Goldenson, and H.-W. Jung, "SPICE in Retrospect: Developing a Standard for Process Assessment," *J. Systems and Software,* vol. 80, no. 9, pp. 1483-1493, 2007.

[82] K. Schwaber, *Agile Project Management with Scrum.* Microsoft Press, 2004.

[83] W.R. Shadish, T.D. Cook, and D.T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference.* Houghton-Mifflin, 2002.

[84] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort Estimation Using Analogy," *Proc. 18th Int'l Conf. Software Eng.,* pp. 170-178, 1996.

[85] M. Sidman, *Scientific Research.* Basic, 1960.

[86] H.P. Siy, A. Mockus, J.D. Herbsleb, M. Krishnan, and G.T. Tucker, "Making the Software Factory Work: Lessons from a Decade of Experience," *Proc. Seventh Int'l Symp. Software Metrics,* pp. 317-327, 2001.

[87] D.I.K. Sjøberg, T. Dybå, and M. Jørgensen, "The Future of Empirical Methods in Software Engineering Research," *Future of Software Eng.,* pp. 358-378, IEEE-CS Press, 2007.

[88] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A.C. Rekdal, "A Survey of Controlled Experiments in Software Engineering," *IEEE Trans. Software Eng.,* vol. 31, no. 9, pp. 733-753, Sept. 2005.

[89] H. Tetens, "Reproducibility," *Enzyklopädie Philosophie und Wissenschaftstheorie,* J. Mittelstrass et al., eds., pp. 593-594, Metzlersche J.B., 2004.

[90] W. Trochim, "Outcome Pattern Matching and Program Theory," *Evaluation and Program Planning,* vol. 12, no. 4, pp. 355-366, 1989.

[91] D.G. Wagner, "The Growth of Theories," *Group Processes: Sociological Analyses,* M. Foschi and E.J. Lawler, eds., pp. 25-42, Nelson-Hall, 1994.

[92] F. Walkerden and R. Jeffery, "An Empirical Study of Analogy-Based Software Effort Estimation," *Empirical Software Eng.,* vol. 4, no. 2, pp. 135-158, 1999.

[93] R.K. Yin, *Case Study Research: Design and Methods,* third ed. Sage Publications, 2003.

[94] R.K. Yin, P.G. Bateman, and G.B. Moore, "Case Studies and Organizational Innovation: Strengthening the Connection," *Science Comm.*, vol. 6, no. 3, pp. 249-260, 1985.

Bente C.D. Anda received the MSc and PhD degrees from the University of Oslo in 1991 and 2003, respectively. From 2002 to 2008, she was a research scientist at Simula Research Laboratory, Lysaker, Norway. She also has several years of industry experience as a consultant. She is currently a senior advisor and group manager at the Norwegian Directorate of Taxes and an associate professor in the Department of Informatics at the University of Oslo. Her research interests include empirical software engineering, software project management, software quality, and software process improvement.

Dag I.K. Sjøberg received the MSc degree in computer science from the University of Oslo in 1987 and the PhD degree in computing science from the University of Glasgow in 1993. He has five years of industry experience as a consultant and group leader. He is a professor of software engineering in the Department of Informatics at the University of Oslo and founded the Software Engineering Department at the Simula Research Laboratory. Among his research interests are research methods in empirical software engineering, software processes, software process improvement, and most aspects of the software life cycle. He is a member of the IEEE, the International Software Engineering Research Network, and the editorial board of *Empirical Software Engineering*.

Audris Mockus received the BS and MS degrees in applied mathematics from Moscow Institute of Physics and Technology in 1988 and the MS and PhD degrees in statistics from Carnegie Mellon University in 1991 and 1994, respectively. He is interested in quantifying, modeling, and improving software development. He designs data mining methods to summarize and augment software change data, interactive visualization techniques to inspect, present, and control the development process, and statistical models and optimization techniques to understand the relationships among people, organizations, and characteristics of a software product. He is with the Software Technology Research Department at Avaya Labs. Previously, he was with the Software Production Research Department at Bell Labs. He is a member of the IEEE.