Software Effort Estimation Error

Stein Grimstad

Thesis submitted for the degree of Ph.D.

Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo

November 2006

Abstract

Software effort estimation is an important part of software development work and provides essential input to project feasibility analyses, bidding, budgeting and planning. The consequences of inaccurate estimates can be severe. For example, estimates that are too optimistic can cause significant losses and those that are too pessimistic can result in contracts being lost. Unfortunately, it is common for software development projects to overrun their effort estimates, typically because the estimates are too optimistic. The average overrun is reported to be 30-40% of the estimated effort.

In this thesis, I focus on expert judgment-based effort estimation. This estimation approach is by far the most frequently applied estimation approach in the software industry. In addition, even when formal effort estimation models are applied, expert judgment typically plays an important role in providing input to the models. In spite of the obvious industrial importance of expert judgment in effort estimation, and the lack of evidence that formal estimation models provide more accurate effort estimates, the main focus of research has so far been on the effort estimation models.

The goal of my thesis is to contribute to reducing the estimation error in software development projects by coming to a better understanding of the shortcomings of, and how to improve, expert judgment-based processes of effort estimation. The following topics are addressed:

- *When, and how much, do judgmental biases and inconsistencies in effort estimates impact the estimation error?*
- *What is the software clients' role in reducing effort estimation error in software development projects*?
- *What is the state of practice of software effort estimation error measurement and analysis?*

The selection of these topics is motivated by their importance for the improvement of expert judgment-based effort estimation error. Accurate expert judgment-based effort estimates will be impossible in the following circumstances: when the estimator is strongly affected by irrelevant information; when the same estimation information leads to widely differing estimates by the same estimator on different occasions; when the clients' behaviour prevents meaningful effort estimation work; when there is no common understanding of what is meant by an effort estimate; and when the interpretation of

estimation error measurements is flawed. An additional motivation is the observation that these topics have previously received scant research attention.

The topics are addressed in three experiments with, in total, 175 software professionals as subjects, one survey with 300 software professionals as respondents, one analysis of 19 completed projects within one software development company, and two reviews of the software estimation literature. Main findings include:

- The presence of estimation irrelevant information as input to expert judgment-based processes of effort estimation can affect the estimates severely. Knowledge and acceptance that the information is irrelevant do not result in the effect being eliminated. This implies that the only safe method of eliminating the effect of irrelevant information is to remove the information before the estimation work starts.
- The level of inconsistency in expert judgment-based estimation processes is surprisingly high. If we do not introduce training and processes aimed at higher levels of estimation consistency, we can hardly ever expect accurate effort estimates.
- Software professionals perceive that clients affect estimation accuracy, especially issues regarding system requirements. This implies that the clients should play an important role in the improvement of estimation processes.
- Imprecise estimation terminology and inappropriate processes of estimation error analysis are common in software estimation research. One consequence is that the results of many estimation error analyses, both in industry and research, are easily misinterpreted. For example, it is often difficult to know in field studies whether an increase in average estimation error indicates a decrease in estimation skill, poorer project control, or higher estimation complexity. I suggest a framework for, what I believe is, better estimation error analyses.

In addition to the findings' individual contributions to the body of knowledge in software engineering, together they contribute to a better understanding of the huge variety of topics that researchers need to address to improve estimation processes. Topics of relevance are related not only to typical engineering processes, but also to such various topics as communication of estimation information (including terminology), clients' estimation behaviour, human biases, and judgmental inconsistency. Therefore, to address the "estimation problem", software engineering researchers need to apply knowledge and research methods from a variety of research disciplines.

Acknowledgements

First, and foremost, I would like to extend special thanks to my supervisor, Professor Magne Jørgensen. I cannot imagine how anyone could have made the Ph. D. process more educational, inspiring or enjoyable.

Thanks to Nils Christian Haugen. Our many and extended discussions have provided valuable input.

Thanks to Erik Arisholm, Hans Christian Benestad, Tanja Gruschke, Thor Henning Hetland, Bente Hoff, Sindre Mehus, Kjetil Moløkken-Østvold, Frode Torvund, and to my other colleagues and friends who have contributed to this thesis.

Thanks to my superiors at Simula and Objectnet: Kjell Nybråten, Dag Sjøberg, Sverre Tinnen, and Aslak Tveito.

Thanks to Chris Wright at Cambridge Language Consultants.

Finally, thanks to the Norwegian Resource Council's SPIKE project for funding.

List of Papers

The following papers are included in the thesis:

[I] **The Impact of Irrelevant Information on Software Effort Estimates**

S. Grimstad and M. Jørgensen

Submitted to 18th Australian Conference on Software Engineering, Melbourne, Australia, April 10-13, 2007

[II] **Inconsistency in Expert Judgment-based Estimates of Software Development Effort**

S. Grimstad and M. Jørgensen

Submitted to Journal of Systems and Software, 2006.

[III] **The Clients' Impact on Effort Estimation Accuracy in Software Development Projects**

S. Grimstad, M. Jørgensen and K. J. Moløkken-Østvold

In 11th IEEE International Software Metrics Symposium (METRICS), Como, Italy, September 19-22, page 3, 2005.

[IV] **Software Effort Estimation Terminology: The Tower of Babel**

S. Grimstad, M. Jørgensen and K. J. Moløkken-Østvold

In Journal of Information and Software Technology 48(4):302-310, 2006.

[V] **A Framework for the Analysis of Software Cost Estimation Accuracy**

S. Grimstad and M. Jørgensen

In 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE), Rio de Janeiro, Brazil, September 21-22, pages 58–65, 2006.

Related Publications

As part of my Ph. D. work I have made contributions to the following papers not included in the thesis:

[VI] **Over-optimism in Software Development Projects: "The winner's curse"**

M. Jørgensen and S. Grimstad

In: Proceedings of IEEE CONIELECOMP, Puebla, Mexico, February 28-March 2. IEEE Computer Society, pages 280–285, 2005.

[VII] **Management of Public Software Projects: Avoiding Overruns**

K. J. Moløkken-Østvold, M. Jørgensen, P. Sørgaard and S. Grimstad

In: Hawaiian International Conference on Business, Hawaii, USA, May 25-28. Hawaiian International Conference on Business, 2005.

Contents

1. Motivation, Research Questions and Contributions

Computer use now pervades almost every aspect of our daily lives. We use computers at work, we use email to communicate with friends and colleagues, and we shop on the internet. Somewhat less visible, but equally important, are the computers that are embedded in other products. For example, cars, fridges, pacemakers, ATMs, and ticket systems would not work without their integrated computers. Central services such as health care, airline traffic, and power plants have been made safer and more efficient by the use of computers. In short, today's society relies heavily on the use of computers.

However, computers need to be told what to do in order to function in ways that are useful to us. People have to write instructions that the computer can interpret. Without these instructions, computers are useless. The instructions are often referred to as "computer programs", "applications" or, perhaps most commonly, "software". When software fails, the consequences can be dramatic. For example, in 1995, 164 people were killed when a passenger aeroplane from American Airlines crashed into a mountain in Colombia. The subsequent investigation revealed that the crash was a direct consequence of an error in the software. In most cases, the consequences of software failure are not fatal, but they are often severe. For example, it is easy to imagine that flawed information can lead to flawed decisions, and consequently, to severe financial losses.

Software engineering is the discipline of designing, creating, and maintaining software by applying technologies and practices from computer science, project management, engineering, application domains and other fields[*]. It can be expensive, complex, and demanding of resources. Consequently, there is a high demand for predictable software development processes that yield high-quality software.

Since the 1960s and 1970s, academia and the software industry have made large investments in software engineering research and development. This has produced considerable insight into the complex domain of software development, and has led to improved tools, languages, methodologies and techniques. We may lack strong scientific evidence of this improvement, but few professionals doubt that we have increased our ability to work more efficiently, and to build more complex software, over the years. However, there is at least one field in which little progress seems to have

[*] See http://www.wikipedia.org (7. Nov. 2006)

been made: software effort estimation. This is unfortunate, because effort estimation is the basis for important software development processes such as bidding, planning, budgeting and determining the project's feasibility.

When we read about "software project failures" in the press, these failures refer, in most cases, to the consequences of inaccurate effort estimates. A recent example is that the U.S. government "lost" $318 on improper tax refunds because a new version of a software system used by the Internal Revenue Service (IRS) that screens tax returns for fraud was delayed[*]. The IRS had counted on the software being ready as estimated and did not have any backup plan when it did not. There are many other examples of severe problems caused by inaccurate estimates, e.g. people who get fired because their projects overrun the effort estimates, companies that encounter financial trouble because their estimates are too pessimistic and therefore lose many contracts, companies that have lost business opportunities because the estimates were too optimistic and the end product had to be delayed, suboptimal resource allocation because the cost benefit analysis were based on unrealistic estimates, and unexpected high software maintenance costs because the quality of the software was compromised in order to reduce estimation errors.

Scientific studies confirm the poor state of software effort estimation. A recent review [33] reports that 70-80% of software development projects overrun their estimates and that average overruns are about 30-40%. These figures seem to be stable over location and time. Studies with similar results have been conducted in, e.g., the USA, Norway, New Zealand and the United Kingdom. Further, there has been little, if any, reported improvement in software effort estimation accuracy since the earliest surveys were conducted more than 20 years ago. The only studies reporting a strong improvement are the Standish Groups' Chaos reports[†]. They claim that in 1994 the average overrun in software development projects was as high as 189%, and that it was reduced to 45% in 2000, i.e. a huge improvement. However, as pointed out in [20], there seem to be several severe methodical shortcomings in the Standish reports. There is an ongoing debate about whether or not these results are trustworthy. As I point out in paper IV, it is not trivial to interpret the exact meaning of the figures reported in the other studies either, but at least it seems quite clear that effort estimation errors are common and that the errors are often rather large.

[*] See http://www.foxnews.com/story/0,2933,211887,00.html (7.nov.2006)

[†] See http://www.standishgroup.com

It is unrealistic to expect perfect effort estimates even with perfect estimation processes, because the effort use is probabilistic by nature, i.e., there are many possible effort outcomes with connected probabilities at the time of estimation. The effort used to implement a software development project depends on many factors, e.g. the amount of implemented functionality, the number of errors made by the programmers, the estimated effort use, and the quality of the code that is produced. Some of these factors can be known upfront, such as the availability of appropriate development tools, while other, probabilistic factors, such as the absence of staff due to illness, cannot. The presence of probabilistic factors means that until the project is finished, the remaining effort use is a distribution and an interval of estimates have a nonzero probability of being equal to the actual effort. Good estimation processes will produce estimates with a higher probability of being correct than less good processes.

Accurate effort estimates are also difficult to achieve because of the complex relationships among factors relevant to effort. As an illustration, some factors cause more use of effort directly, e.g. unexpected problems configuring the development tools; some factors may lead to other events and cause more use of effort indirectly, e.g. use of new and untested development tools; some factors enable other factors to sometimes lead to more use of effort, e.g. use of a development methodology that depends on a new development tool; and yet other factors do not cause more use of effort but increase use of effort if not in place, e.g. availability of skilled development tools support personnel.

The probabilistic nature of effort usage in software development projects and the complex relationships among factors relevant to effort mean that we should not expect an easy solution to the estimation problem. However, given the current state of highly inaccurate software effort estimates and the sometimes dramatic consequences of inaccurate estimates, we must continue to search for ways of improving effort estimation. Even changes in estimation methods that lead to small improvements in estimation accuracy may be highly profitable, due to the amounts of resources used by, and the potentially damaging consequences of, poorly planned software projects. The strong need for improved estimation accuracy is a major motivation for the choice of effort estimation as the topic of this thesis.

The remaining part of this section describes and motivates the research questions (Section 1.1), provides a summary of the main contributions (Section 1.2), and describes the structure of the thesis (Section 1.3).

1.1. Research Questions

The approaches to solve the complex problem of estimating software development effort are typically classified as expert estimation or formal estimation model-based. The term "expert estimation" is typically used as a label for estimation methods in which a significant part of the estimation process (particularly the final step, i.e., the “quantification step”) is based on intuition, while the term "formal estimation models" is typically used as a label for estimation methods where a substantial part of the estimation (and particularly the "quantification step") is based on the use of mechanical and analytical processes, e.g., the use of a formula derived from historical data using regression analysis. Software effort estimation research is inconclusive regarding which estimation approach is better, e.g. a recent review [17] of studies comparing models and experts in software development effort estimation concludes that experts typically performs no worse than the models.

Reduction of estimation error has been a major concern in the software engineering community for a long time. Most of this research has focused on the development and evaluation of formal estimation models. However, despite a massive amount of research on formal estimation models [19] and recommendations on the use of estimation models in software engineering textbooks and process improvement frameworks, expert judgment-based effort estimation is by far the most popular estimation method in the software industry. This may be due the lack of convincing results regarding the accuracy of estimation models, but may also be due to the fact that even estimation models rely on expert judgment to provide some of the input.

Motivated by the dominant use of expert estimation in the software industry, the lack of convincing results regarding the benefits of switching to formal estimation models, and the importance of improved judgmental processes even when using formal estimation models, I chose to focus my research on the improvement of judgment-based software development effort estimation processes. The overall research question that is investigated in thesis is therefore:

RQ: How can expert-judgment based effort estimation processes be modified to reduce software effort estimation error in software development projects?

There are many ways of improving judgment-based effort estimation processes. In my thesis I focus on the following means of improving effort estimation accuracy:

- Coming to a better understanding of why and how irrelevant information affects effort estimates, and of the level of, and reasons for, inconsistency in effort estimates.
- Coming to a better understanding of the client-related factors that can cause and prevent estimation errors in software development projects.
- Developing better processes for analysing estimation error, including our own analyses.

These means were the starting point when formulating the research questions (see Table 1) addressed in the studies. The underlying assumption is that a better understanding of important relationships, e.g., the relationship between irrelevant information and estimation accuracy, is likely to promote the development of better estimation processes.

Table 1 Research Questions and Motivations

Research Question	**Motivation ("Research gaps")**
RQ 1: How can biases and inconsistencies in effort estimates based on expert judgment be reduced?	Few software estimation studies address expert judgment-based estimation processes [19] and we have a poor understanding of the cognitive processes involved [16].
RQ 1.1: When, and how much, are software professionals' estimates of most likely effort impacted by estimation-irrelevant information in the requirement specifications?	Several studies report that poor requirement specifications can increase software effort estimation error, e.g. [26, 43]. Currently, we know little about the role of estimation irrelevant information in specifications on the effort estimates. Research from other fields, e.g. [36, 46], suggests that such information in the input to forecasting processes could be an important cause of inaccuracy in forecasting processes that relies on human judgment.
RQ 1.2: How consistent are software professionals' expert judgment-based effort estimates?	Several studies show that the variance of effort estimates when different estimators estimate the same task is large, see e.g. [24], but I am not aware of any studies that address the topic of intra-subject consistency in a software estimation context, i.e., there have been no previous studies on how consistently the same software professional would estimate the same project based on the same information on different occasions. Research from other fields typically finds that judgmental forecasts are inconsistent, and that this inconsistency is a major source of error that contributes to reduced forecasting accuracy and makes learning more difficult; see e.g. [41].
RQ 2: What is the software clients' role in reducing effort estimation error in software development projects?	Client-related factors are believed to affect estimation error, as reported in e.g. [38, 39]. The results reported in [34], for example, suggest that government projects had higher overruns of effort estimates than private projects. Despite the apparent importance of client-related factors, there has been little, if any, systematic investigation of such factors.

RQ 3: What is the state of practice of software effort estimation error measurement and analysis and how can it be improved?	Flawed measurement and analysis of estimation error can lead to flawed conclusions. While there has been some research on the estimation error measures themselves, e.g., on properties on the MMRE-measure [40], I have not found much research on other important steps of estimation error measurement and analysis, e.g., studies on how to ensure that what we measure is estimation skill and not something else.
RQ 3.1: Is the term 'effort estimate' precisely defined in the software engineering literature?	Several papers demonstrate the importance of a precise estimation terminology for effort estimation measurement, e.g. [14]. However, no studies I am aware of have investigated the state of estimation terminology in the software industry and research, i.e. to what degree there is a need for improved use of terminology and what the main problems are.
RQ 3.2: What is the state of estimation error analysis, and when does the lack of proper estimation error analysis bias analyses of effort estimation accuracy and, in the worst case, lead to incorrect conclusions?	Several general frameworks for analysing software measurements exist, e.g. the GQM (Goal-Question-Metrics) framework. These frameworks for analysis are neither specific nor tailored to software estimation. Consequently, there may be a need for additional frameworks for analysis, based on knowledge about the shortcomings of current analyses of estimation error.
	.

Further motivation for the selection of research questions is provided by the comprehensive review of software estimation research papers reported in [19]. Based on the review of 304 software effort estimation papers, the authors argue for more research in six areas. I believe the research questions I have selected respond to at least the following two recommendations:

1) *"… current research on, e.g., expert estimation is sparse and we believe that it deserves more research effort."* ([19], p. 22)

2) *"… we should put much more research effort into developing better ways of evaluating estimation methods and of measuring the accuracy of estimations."* ([19], p. 22)

Table 2 shows how each paper in the thesis relates to the research questions.

Table 2 Research Questions and Papers

Research Question	**Paper Id**	**Title of Paper**
1.1	I	The Impact of Irrelevant Information on Software Effort Estimates
1.2	II	Inconsistency in Expert Judgment-based Estimates of Software Development Effort
2	III	The Clients' Impact on Effort Estimation Accuracy in Software Development Projects
3.1	VI	Software Effort Estimation Terminology: The Tower of Babel
3.2	V	A Framework for the Analysis of Software Cost Estimation Accuracy

1.2. Main Contributions

The results presented in this thesis provide empirical evidence that the error of judgment-based effort estimates can be reduced by:

- Removing irrelevant information from the requirement specifications prior to estimation.
- Practices that reduce inconsistency, e.g., mechanically combining independent estimates from several estimators.
- Developing greater awareness of the clients' role in promoting realistic effort estimates.

The evidence in support of these recommendations are based on the following:

- The results from two experiments presented in Paper I that suggest that the presence of estimation-irrelevant information in input to expert judgment-based effort estimation processes can severely hinder the estimators' ability to consider only relevant information, and thereby lead to inaccurate estimates. For example, the presence of irrelevant information increased average effort estimates by 30% in one of the experiments.
- The results from an experiment presented in Paper II that suggest that there is a surprisingly high level of inconsistency among estimators, e.g. in the experiment the subjects estimated identical tasks on several occasions and the mean difference of the effort estimates of the same task by the same estimator was as much as 71%. This high level of inconsistency may be a major source of error in software effort estimation. In addition, inconsistent use of estimation information and processes may reduce the realism of the effort estimates and hinder learning.
- The results from a survey presented in Paper III that suggest that software professionals believe that clients affect estimation accuracy. Changed and new requirements are perceived as the clients' most frequent contribution to overruns, while overruns are prevented by the availability of competent clients and capable decision makers.
- The results from two reviews and an analysis of completed projects within one software development company, presented in Paper IV and Paper V, suggest that imprecise estimation terminology and inappropriate estimation error analysis processes makes it in many cases difficult, if not impossible, to interpret and make meaningful use of the results of studies that report and analyze software effort estimation error, e.g. we found that only two out of 23 reviewed research papers used

a precise estimation terminology. A framework that, I believe, enables a better process for analysing estimation error is suggested.

These findings indicate that a wide range of topics, as different as estimation terminology, cognitive processes and client-related factors, can have an impact on estimation errors. Consequently, it seems necessary to conduct studies of many types, and from many viewpoints, in order to reduce estimation error. This thesis focuses on improving expert judgment-based effort estimation processes, but I believe that the results regarding inconsistency and biases may be useful input to the improvement of any estimation method that is based, at least in part, on human judgment, and that the results on estimation error analysis and the customers' impact on estimation error, may be valuable for an even broader audience. However, in order to make a substantial contribution to the reduction of software effort estimation error, we may need to develop theories that explain the processes of expert judgment better. This thesis does not provide such theories, but the findings presented will, I hope, contribute to their development.

I have not found any studies that were published after the papers in the thesis had been written that support, explain or contradict my findings. This is, perhaps, unsurprising, given the short time span between the writing of the papers and the writing of the thesis.

1.3. Overview of the Thesis

This work is organized as follows. Section 1 (this section) introduces the topic of software cost estimation and motivates the need for more research in this area. The section presents and motivates the research questions investigated and describes the main contributions related to each research question. Section 2 discusses the state of software effort estimation practice and research, with a focus on topics relevant to the research questions. Section 3 briefly presents alternative research methodologies in empirical software engineering, and provides an overview of, and motivation for, the research methodology decisions made in this thesis. Section 4 summarizes the results of the individual papers. Section 5 contains a summary of my contributions to the idea, design, data collection, analysis and writing of the individual papers included in the thesis. Section 6 presents further work. Section 7 provides some concluding remarks. The five individual papers are included at the end of this thesis.

2. Software Effort Estimation Error

Software estimation has been recognized as a problem since the early days of software development; see, e.g. the report from 1965 reported in [9]. A considerably body of research has been produced. In this section, I define and explain what we mean by “effort estimate” in this thesis. Then, I briefly describe related research results from three research areas: 1) estimation methods, 2) causes for estimation error, and 3) measurement and analysis of estimation error. These results are intended to provide background material for the discussion and understanding of the studies reported in the thesis.

2.1. Terminology

The term *software development effort estimate* is understood as a prediction of the effort most likely required to implement a software development project. An effort estimate is produced by an estimation process that takes estimation-relevant (and sometimes irrelevant) information as input and translates it into the effort estimate or a distribution of effort values with connected probabilities. As described in Section 1, this process may be dominantly mechanical (model-based) or judgmental (expert estimation-based). An effort estimate is based on a set of implicit or explicit assumptions, e.g. the assumption that the implemented system will be the same as that which is described in the requirement specification. Some assumptions are made by the people who estimate the project, while others are constraints that are enforced on the project, e.g. that the project has to be delivered within three months.

All estimates are uncertain. This uncertainty can be expressed as confidence intervals, e.g., 90% confidence minimum-maximum intervals, or more informally, e.g., that an estimate is likely to be quite inaccurate. Typically, the uncertainty of an estimate decreases when more information is available.

2.2. Estimation Methods

A plethora of estimation methods exists and there are several schemas for classifying them, e.g. [4, 21]. In what follows, I discuss estimation methods on the basis of the classification "expert judgment" and "formal estimation models". This is similar to the top level of the categorization schema used in [5] (model-based estimates vs. non-model based estimates).

Formal estimation models

There are many different types of formal effort estimation models available. Many of these models are categorized and described in [5]. Effort estimation models may be based on sophisticated analyses and dependencies between effort and other variables in sets of previously completed projects and result in, e.g., formulae of the following type:

$$Effort = a\ Size^{b} * Adjustment\ factor$$

The Size variable can, for example, be a measure of the ‘size of functionality’ derived from the requirements specified by the client or the estimated number of lines of code to be programmed. The adjustment factor is typically derived from a weighted sum of answers to questions related to development complexity, project member skills and tools used to support the development process. The adjustment factor may also include input of a productivity factor, i.e., a measure of the historical productivity of similar projects. Examples of well-known estimation models are these:

- COCOMO [3]. In COCOMO, the use of effort of a typical project is assumed to follow a pre-defined formula regarding the relationship between Size and Effort. To account for differences in productivity, it is possible to make a number of (to some extent subjective) adjustments to the “nominal” effort estimate, e.g., the estimate can be adjusted for level of reuse.
- Use Case Point Estimation (UCP) [1]. The method takes as main input a software specification described through “Use Cases”, which is part of the notation of UML [25] and is similar to the use of “Function Point”-based estimation models. Each use case is counted and weighted, and is adjusted for technical parameters. An expected productivity rate is provided as input.

Current research in this area is typically directed at making better models, e.g. [27], investigating when, how and how much local calibration of the models that are beneficial, e.g. [28], and evaluating models, e.g. [31].

Model-based effort estimation processes may rely on expert judgment-based input. Hence, model output may also be biased towards over-optimism or be affected by the presence of irrelevant information. Further, inconsistency in effort estimates may be a problem, despite the use of models. The fact that model-based estimation processes sometimes rely on expert judgment-based input illustrates the relevance of studies on expert judgment-based estimation, even when models are used.

Expert judgment-based estimation

Expert judgment-based estimation is not a single estimation method, but a spectrum of judgment-based processes. Even within the same software development organization, processes of expert judgment-based estimation may vary from those based on pure intuition to highly structured processes that use relevant historical data, in which it is principally the essential step from understanding the software development estimation problem to quantifying the required effort that is based on intuition. There are several estimation processes and support material designed to aid expert judgment-based estimation, e.g.:

- "Planning Poker" (PP) [6], which is a set of estimation process guidelines that are rapidly gaining popularity in the Agile community. PP is based on the combining of anonymous, individual estimates from the developer team and on group discussions.
- Work Breakdown Structures (WBS) guidelines, e.g. [42]. A WBS typically involves a template for partitioning software project work into smaller tasks that are estimated separately. These estimates are combined to provide the total estimate.
- Estimation checklists. These checklists may be company-specific and based on, e.g., the company’s perception of activities that are typically forgotten when estimating.

Current research on expert judgment-based estimates is typically directed at discovering faults in the process of human judgment that bias the estimates, e.g. [2], developing and evaluating methods and training processes that reduce the impact of estimation biases, e.g. [44], and the development of models that aim at a better selection of realistic estimators, e.g., [16].

As can be seen from the above discussion, expert estimation can include structured and analytical processes. The difference from model-based effort estimation is sometimes not large, and mainly due to differences in the “quantification step”, i.e., the step from understanding what to do to how much effort it will require.

Which approach is better?

Surveys of industry practice have shown that expert judgment-based estimation is by far the most commonly used estimation approach, e.g. [11, 12, 37]. In spite of this, only 15% of the papers on estimation methods address the use of expert judgment [19], and then typically as a means to evaluate the performance of a recently developed estimation model.

Studies are inconclusive regarding whether we can expect more accurate estimates with models or expert judgment. The review published in [17] reports on 16 studies that compare expert judgment-based estimation with formal estimation models. That review shows that the average accuracy of expert judgment-based effort estimates was greater than the average accuracy of the models in 10 of the 16 studies. The lack of systematical superiority of either approach might be explained by the fact that most meaningful models are at least partly based on expert judgment, i.e. expert judgment is used to produce the input to the models.

These results are somewhat surprising. Compelling evidence from other fields shows that expert judgment is, in most cases, outperformed by even the simplest estimation models; see, e.g. [30]. Whether this is because the models used in software estimation are of low quality, or because the complexity of software estimation is high, or because there is some other problem, is unknown. Some studies have investigated the combination of methods, exploring the advantage of both, e.g. less bias in models and flexibility in expert judgment, with promising results.

In relation to the studies presented in this thesis it is important to note that one reason for the difficulty in comparing expert judgment and model-based effort estimation is the lack of understanding of conditions for good and poor estimation performance by experts, i.e., when we can expect software professionals to produce more accurate estimates than the estimation models. I believe that the results reported in this thesis represent a step in that direction, e.g., in that my studies report that the availability of a high amount of irrelevant and misleading information indicates poor expert judgment.

2.3. Factors that Affect Estimation Error

In order to improve estimation processes, it is necessary to understand why and when software cost estimation errors occur. A better understanding of such factors is important input to many activities relevant to the reduction of software effort estimation errors, such as developing and improving estimation methods, analysing estimation errors, determining productivity factors, and choosing development methods. Previous research on this topic relies mainly on interviews/surveys of project members and statistical analysis of the recorded characteristics of completed projects.

In [18] it is shown that the data collection approach, who you ask and the methods used for analysis, can provide quite different results regarding the causes of estimation errors for the same projects. However, it would seem that these results from different data collection approaches and analysis methods are complementary rather than contradictory, i.e., the importance and descriptions of estimation error factors are related strongly to perspectives.

The transfer of the causes of estimation error from one organization or project to other organizations or projects is not trivial. For example, the level of maturity of the clients may be an important factor regarding estimation error in one organization, while organizations that have mostly mature clients will not perceive client maturity as a problem. A further complicating factor is that the nature of software development projects changes continuously, e.g. in many modern web applications, deployment issues are less of a problem than in many traditional client-server applications. Hence, previous results are not of obvious value for current projects.

The organization and project dependency of estimation-relevant factors, together with changes over time regarding the importance of these factors, means that the value of the survey-based studies on estimation errors, e.g., studies based on questionnaires sent to a large number of project leaders in organizations of various sizes and project types, may be questionable. This problem with the current studies on factors regarding estimation error motivates our focus on establishing a framework that will hopefully provide good support for software organizations to conduct their own analyses on their own data.

2.4. Effort Estimation Error Measurement

Measuring estimation error is a fundamental activity in software estimation research. It is the basis of many activities, such as analysing whether or not an organization has an estimation problem, evaluating estimation methods, and identifying causes of estimation error.

Software projects differ in size and, for aggregation purposes, most measures of estimation error are consequently based on relative estimation error. The most commonly used measure of estimation error is the Magnitude of Relative Error (MRE) [7]. MRE is calculated by the following formula:

$$MRE = \frac{|\,\mathit{actual\ effort} - \mathit{estimated\ effort}\,|}{\mathit{actual\ effort}}$$

The mean MRE (MMRE) is often used to average estimation error for multiple observations. However, MMRE can be very sensitive to extreme values, and it is therefore sometimes preferable to use median MRE (MdMRE) instead. A related metric is PRED, which is a measure of how many observations lie below a specified level of estimation error (MRE), e.g., PRED(25%) is a measure of the proportion of estimates that deviate less than 25% from the actual effort.

It is not unproblematic to use MMRE as a measure of estimation accuracy; see [10, 40] for discussions of problems related to, among other things, a correlation between MRE and project size and the implicit loss function that penalizes effort overruns less than effort underruns. The use of a measure of the relative estimation error has, in addition, the disadvantage that small projects may easily be given too much weight in the MMRE, e.g., a 10 man-year project with 20% effort overrun contributes as much to the MMRE as a 10000 man-year project with 20% effort overrun.

MMRE measures estimation error relative to the actual effort. In many cases, it is more interesting, and perhaps more intuitive, to measure estimation error relative to the expected effort, i.e. the estimate, as done by MER (the magnitude of error relative to the estimate) [23]:

$$MER = \frac{|\,\mathit{actual\ effort} - \mathit{estimated\ effort}\,|}{\mathit{estimated\ effort}}$$

A problem with both MER and MRE is that they place uneven weight on over- and underestimation. A more balanced metric is BRE [32], calculated by

$$BRE = \frac{|\,\mathit{actual\ effort} - \mathit{estimated\ effort}\,|}{\min(\mathit{actual\ effort}, \mathit{estimated\ effort})}$$

Other, less used, measures include the relative standard deviation (RSD), the logarithmic standard deviation (LSD). In other fields, similar measures may have other names; see, e.g. [29]. However, all estimation error measures have shortcomings [10, 13]. Hence, the measure that should be used in any given case depends on the context [35]. The studies included in the thesis mainly use MRE as the measure of estimation error. I believe that this choice is acceptable, given the similarity of the tasks' size.

Factors other than the choice of estimation error measure can have a large impact on the validity of a measurement of estimation error. An obvious example is the lack of clear terminology for effort estimation. Although all the presented measures are based on "estimated effort", it is not clear how to interpret this term in most field settings. In [8], for example, the authors found that the term "effort estimate" was used as a substitute for concepts with such different purposes as the following: most likely effort (Purpose: Realism), planned effort (Purpose: Project control), budgeted effort (Purpose: Budget control), price (Purpose: Winning a bidding round), and combinations of these concepts. In [15] it is argued that mixing these concepts and purposes leads easily to less focus on realism. In addition, the way in which estimation error is interpreted obviously depends on the intended meaning of the term “effort estimate”.

I noted above that many factors can affect estimation error. Measuring estimation error without a clear understanding of which factors contributed most to the estimation error, e.g., without an understanding of whether a high estimation error is caused by the factor “low estimation ability” or “high estimation complexity”, is rarely meaningful.

These problems related to the imprecise use of estimation terminology and the isolation of estimation error factors, are in my opinion, perhaps even more important to solve than the selection of the optimal estimation error measure, and have motivated my selection of research topic. These topics have received little attention in research on software estimation.

3. Research Methods

Section 3.1 briefly describes the use of some common empirical research methods in software engineering. For a more thorough discussion of these methods and related issues see, for example, [45]. Section 3.2 gives an overview of and motivates the important decisions of this thesis that pertain to research design.

3.1. Research Methods used in Empirical Software Engineering

Experiments

In an software engineering experiment, the subjects are typically randomly assigned to a treatment. Then they perform one or more tasks and the effect of the treatment is measured. The effect of the treatment is then analysed statistically, and any differences is assumed to be caused by the treatment. Experiments are typically conducted in a laboratory setting, but may also be conducted in field settings.

The main strengths of experiments are that they enable a high level of control, in that the effect of the studied factor can be isolated, and that experiments are typically easier to replicate than other types of study. The main weakness is that the artificiality of the tasks and contexts may make generalizations, particularly of the effect sizes, to field settings difficult.

Observational studies (single- or multiple-case studies)

In a case study, one or more projects or activities are typically observed. Data may be collected in several ways, e.g. by interviewing project members and by analysing code. Observational studies are typically conducted in a real-life context, e.g. the researcher observes a software development project in an organisation.

The main strength of observational studies is that they enable the researcher to study a phenomenon in a realistic context with rich data. The main weaknesses are the lack of control, e.g. it can be difficult to isolate the impact of a particular factor, and the fact that they are difficult to replicate.

Surveys

In a survey, data about a phenomenon is collected from many sources using questionnaires or interviews. Data is typically collected in a standardized way and from a defined population, e.g., from software organizations within a country. Surveys are typically performed after the phenomenon to be studied has occurred, e.g. when a project has been completed.

The main strengths of surveys are that they are easy to replicate and that the cost of investigation is low; hence, a large number of subjects can be included and many factors can be measured. The main weaknesses are that it might be difficult to interpret the answers, e.g. the answers reflect what the subjects believe and what they believe might be different from how things actually are, and that the researcher's level of control is low.

Systematic Reviews

In a systematic review, the existing literature is systematically searched, evaluated and synthesized for evidence regarding a phenomenon or research question. Reviews are often performed prior to other studies in order to establish a background framework and/or to identify gaps in existing research. However, they can also be independent studies, e.g. to test hypotheses.

The main strengths of reviews are that they investigate the effects of some phenomenon over many contexts and research methods, and that data from several studies can be combined in powerful analysis. The main weaknesses, particularly in the context of software engineering, are perhaps that the amount of related research can be low, and that it can be hard to extract data from studies due to non-standardized reporting.

As can be seen, there is no obvious best choice of research method. The choice of research method depends on, among other things, the following: available resources, the need for control to isolate factors to enable generalization by theory, the need for realism to enable generalization-by-similarity or statistical means, the need for in-depth, rich data to understand complex relationships, and the focus of people’s perception rather than the real relationships. My choice of research methods is based on several of these decision elements. In particular, the need for control to isolate relevant factors led to the choice of controlled experiments for two of the papers. Essential decisions regarding design choice are discussed in greater depth in the following section.

3.2. Research Design Decisions

Tables 3-5 give an overview of the research methods applied in the papers included in this thesis. The columns "Advantages of study design" and "Disadvantages of study design" briefly summarize arguments for and against the chosen research design.

Table 3 Overview of Study Designs (Experiments)

Design element	Paper I	Paper II
Main purpose	Study the impact of irrelevant information on expert judgment-based effort estimates.	Study the inconsistency in expert judgment-based estimation processes.
Type of study	Controlled experiment.	Controlled experiment.
Subjects	Experiment 1: 76 software professionals. Experiment 2: 92 software professionals.	Seven highly skilled, software professionals selected based on previous estimation accuracy performance.
Tasks	One small software development task (different tasks in the two experiments).	60 small software development tasks.
Context	The subjects, unaided, estimated the experimental task as an interactive part of a seminar talk (both experiments).	The subjects were paid to come to our laboratory and estimate various enhancements to an existing system. The system documentation was made available to them, and they had previously worked on the system.
Advantages of study design	Realistic (although small) tasks. Software professionals as subjects. High number of subjects. High level of control.	Realistic (although small) tasks. Software professionals as subjects. Previously collected information about the subjects was available. High level of control.
Disadvantages of study design	Only one tasks. Unrealistic estimation environment.	Few subjects. Limitations in realism of estimation environment, e.g., no access to historical data and source code. Expensive.

Table 4 Overview of Study Designs (Surveys)

Design element	Paper III
Main purpose	Investigate how client-related factors are perceived to impact effort estimation error.
Type of study	Questionnaire-based survey.
Subjects	300 software professionals.
Tasks	22 questions regarding client-related factors an how they impact estimation error.
Context	The survey was handed out to participants at an industrial software conference.
Advantages of study design	Software professionals as subjects. Investigates many factors. Inexpensive.
Disadvantages of study design	The data reflects the subjects' explanatory models, and not necessarily the underlying cause-effect models. Possible sample biases (e.g., some projects, clients and companies might be the reference of several responses)

Table 5 Overview of Study Designs (Reviews)

Design element	**Paper IV**	**Paper V**
Main purpose	Assess the state of practice of effort estimation terminology usage as background for proposed improved use of terminology.	Assess the state of practice effort estimation error analysis, and investigate consequences of improper analysis as background for a proposed framework for improved estimation error analysis.
Type of study	Literature review.	Literature review. Analysis of software projects.
Studied elements (subjects)	Software engineering text books. Software estimation research papers.	Review: Software estimation research papers. Projects: 19 software development projects.
Study focus (tasks)	Identification of the use of estimation terminology.	Review: Analysis of how the research papers isolated the studied factor in their estimation error analyses. Projects: Comparison of estimation error of two methods.
Context	Two researchers reviewed the most popular text books and research papers that report estimation error. The review was based on the guidelines in [22].	Review: Two researchers reviewed research papers that report estimation error. Projects: Analysis of completed projects in a software development organization.
Advantages of study design	Good control of publication selection process. Explicit steps make the analyses possible to replicate.	Good control of publication selection process. Explicit steps make the analyses possible to replicate. Enables a demonstration of potential consequences of improper estimation error analysis in a real-life context.
Disadvantages of study design	Possible researcher bias in analysis and selection process.	Possible researcher bias in analysis and selection process.

Considerations that led to my design decisions include the following:

- In Papers I and II, I considered studying the impact of irrelevant information on, and the level of inconsistency of, effort estimation of large development tasks, instead of the rather small programming tasks actually used. Studying large development tasks may have produced different results more applicable to industrial contexts, but would i) lead to a much lower number of subjects and tasks studied due to restrictions on resources, and, ii) increase the risk of introducing “noise” into the experiment, with the consequence that the results would be less reliable. As an illustration, the variation in how a requirement specification is interpreted increases with the number of requirements. The effect of the studied factor, e.g., irrelevant information, would consequently easily have become diluted and would not have been possible to study without a high number of observations if we had insisted on larger requirement specifications. The use of smaller, although realistic, tasks means that the studies point mainly to the existence of the studied phenomenon, and not so much on how large the effect sizes are in an industrial context. However, in Paper II I argue that there are reasons to believe that the level of inconsistency is larger in real projects that the one I observed, i.e., that my observations were made in a consistency-friendly environment.

- In Paper II, when studying the level of and causes of inconsistency, I considered the use of an inter-subject design. This would have been much cheaper than the relatively costly intra-subject design we chose. However, in a inter-subject design it would be difficult to isolate the individuals' level of inconsistency, and, for that reason we decided to use the intra-subject design.
- In Paper III, when studying factors pertaining to estimation error that are client-related, I considered conducting a multi-case study instead of a survey. This was motivated by the belief that the cause-effect relationship under consideration might be insufficiently understood by the survey respondents. Many factors can affect estimation accuracy in software development, and extensive analysis may be required to identify the root causes of estimation error. In surveys such as the one I completed, the respondents have a limited opportunity to perform complex analysis, and consequently their perceptions might be misleading. A multi-case, observational study would have enabled a more in-depth study of the reasons for estimation error and allowed us to triangulate results. The choice of a questionnaire-based survey was based on two factors: 1) the lower cost of the study, 2) the fact that it would better enable us to make comparisons of the reasons contributing to overruns and the reasons preventing overruns with other studies. In hindsight, I find that questionnaire-based studies on complex relationships are very difficult to conduct properly and would not recommend this unless it is the main goal of the survey to monitor the respondents' perceptions. The experiences related to analysis of the design and analysis of the questionnaires motivated, to some extent, the studies described in Papers IV and V, and the research designs in Papers I and II.
- In Papers IV and V, in which properties of research papers and textbooks are reviewed, I considered using external assessors to classify properties of the papers and textbooks. Instead, all classifications were conducted by the authors of the paper. Some of the classifications may be quite subjective and other assessors may have classified differently. The main reason for not using external, independent assessors was practical. It is not easy to find highly skilled assessors for this purpose.
- In Papers IV and V, I considered including material from a wider selection of sources, because there may have been a bias in the selection of publications. For example, both papers include only

publications that address software effort estimation. This means that related material, such as the literature on general project management and forecasting, where a more precise terminology may be present, was not reviewed. I considered this, but on the basis of the impression that such literature was little read by most software professionals and researchers, I decided not to include material from those sources.

4. Results

This section contains a summary of the individual papers that are included in this thesis.

4.1. Paper I

This paper reports on two controlled experiments that investigated the impact of irrelevant information on software development effort estimates. In both experiments, I gave one group of software professionals a requirement specification in which I had included effort-irrelevant information, while I gave the other group the same requirement specification but without the irrelevant information. Seventy-six professional software developers participated in the first experiment and 92 in the second.

I found that information in requirement specifications that is irrelevant to estimation can strongly affect software effort estimates. The average effort estimates increased significantly in both experiments when irrelevant information was included. In addition, the results of the first experiment suggest, somewhat ironically, that estimators may also become more confident in the accuracy of their own estimates when they are exposed to irrelevant information and, consequently, estimate less accurately. The results suggest that estimation accuracy can be improved if information that is irrelevant to estimation is removed from the requirement specifications before being presented to the estimators[*].

More research is needed if this phenomenon is to be understood fully, particularly when and why it occurs, but I believe that the results are rather robust in the sense that they show that irrelevant information may have a large impact on effort estimates

[*] In a recently conducted (as yet, unpublished) study of 172 software professionals I found that the effect of irrelevant information was not eliminated by asking the estimators to identify and highlight (with a yellow pen) the relevant text of a requirement specification. By contrast, removing the irrelevant text by using a black pen, so that it could not be read, reduced the impact of the irrelevant information, although not completely. This suggests that, preferably, a person other than the estimator should remove the irrelevant information. If that is not possible, the "black pen" method of removing irrelevant text should be applied. This result supports and extends the results presented in Paper I.

4.2. Paper II

This paper reports on a controlled experiment that investigated the degree of inconsistency in expert judgment-based estimates of software development effort. Seven experienced software professionals estimated 60 software development tasks, each over a period of three months. Each participant estimated six of the tasks twice. These tasks provided input to my analysis of the level of inconsistency of expert judgment-based effort estimation.

I found that there was a high degree of inconsistency in the participants' effort estimates of the same task. The mean difference between estimates of the same task by the same participant was as high as 71%! The level of inconsistency did not, in my experiment, reduce with increase in task size, i.e., we should not expect the level of inconstancy to decrease with more realistically sized estimation tasks, but rather the opposite. Clearly, effort estimates cannot be very accurate when the degree of inconsistency in the estimates is so large. A consequence of my findings is that it is likely that estimation accuracy can be improved substantially by the use of methods that improve consistency, e.g. by combining effort estimates from several independent estimators.

As with the phenomenon reported in Paper I, this phenomenon is poorly understood and further research is needed. The main finding may be that the level of estimation inconsistency can be surprisingly high among highly skilled software professionals. There is consequently, a need for a stronger focus on improving the consistency of estimation processes to reduce estimation error.

4.3. Paper III

This paper report on a survey conducted at an industrial software engineering conference. The survey contained 20 multiple choice questions and two open-ended questions related to the positive and negative ways in which clients affected the accuracy of effort estimation in software development projects. 300 software professionals participated in the survey.

The respondents answered that the most common client-based factors that produce estimation inaccuracy are the following: (i) frequently changing, and new, requirements, (ii) a lack of well-defined requirements, and (iii) an absence of competent customers and capable decision makers. The most important factors contributing to the prevention of overruns were (i) competent customers and capable decision makers, (ii) adequate project administration and steering, and (iii) well-defined requirements. An additional important finding is related to how difficult it is to design and interpret results from questionnaires about cost overrun factors.

There are several methodological weaknesses with this study, as discussed in Section 3.2. However, in spite of the methodological weaknesses, the results may be a useful starting point for further research on client-related factors that affect estimation accuracy.

4.4. Paper IV

This paper contains a structured review of software estimation terminology in software engineering text books and research papers. I investigated eight software engineering text books and 23 software cost estimation research papers to determine the degree to which the term "effort estimate" was precisely defined, and the degree to which the estimated and the actual efforts were comparable when evaluating estimation accuracy.

Imprecise terminology regarding concepts pertaining to estimation was typical in software engineering text books and research papers. This lack of clarity and precision in the use of terms pertaining to estimation has the following unwanted effects: it makes results regarding estimation accuracy more difficult to interpret; it makes the communication of estimates difficult; and it makes it more difficult for estimators to learn from past estimates. Guidelines for a more consistent terminology, aimed at practitioners and researchers, are suggested.

Despite the potential shortcomings of the review (see Section 3.2 for a discussion), I believe that the following conclusions are quite robust: 1) currently, terminology for estimation is used imprecisely, and 2) the measurement and analysis of estimation error will often benefit from a more precise terminology.

4.5. Paper V

This paper investigates the importance of proper processes for analysing estimation error. To investigate this research topic, I conducted an examination of the analysis of estimation error in a selection of estimation research studies and a study of completed projects. I investigated eight software cost estimation research papers to determine the degree to which strategies to isolate the effects of different estimation error factors were used when analysing estimation error. I also examined the analysis of effort estimation error conducted in a software development organisation.

I found that the lack of proper estimation error analysis can easily lead to flawed conclusions. Few research papers report any attempt to isolate the effects of different estimation error factors. This means that many studies analyse cost estimation error without being able to interpret it properly; without, for example, understanding whether a high estimation error is caused by poor estimation ability, high project complexity, or delivery of more functionality than was assumed when estimating the effort. The results indicate that studies often ignore the potential impact of factors that were not studied, e.g., how systematic differences in estimation complexity or differences in the measurement process can disturb an analysis of the estimation ability. Based on these results, I propose a practical framework for what, I believe, constitutes a sound analysis of estimation error.

In spite of the difference in research questions and expert estimation topics, I believe that all papers contribute to the goal of better understanding the nature of expert judgment-based effort estimation and to the variety of elements that needs to be addressed in order to produce accurate effort estimates. Together, the papers both establish a basis for better analysis of effort estimation error and increased understanding of the variety of reasons for estimation error in software development projects.

5. My Contributions to the Papers

All the five papers that are included in this thesis were written together with my supervisor. Table 6 shows my contribution to each paper.

Table 6 Contributions to Papers

Paper	Idea	Study Design	Data Collection	Analysis	Writing
I	Responsible	Responsible	Responsible	Responsible	Responsible
II	Responsible	Responsible	Responsible	Responsible	Responsible
III	Responsible	Responsible	Responsible	Responsible	Responsible
IV	Shared	Responsible	Responsible	Responsible	Responsible
V	Contribution	Shared	Shared	Shared	Responsible

In addition, I took part in the study design and the analysis in the first related paper (Paper VI) and in the data collection and the writing of the second related paper (Paper VII).

6. Further Work

The findings in this thesis leave many questions unanswered, and there is consequently a need for further research. I intend to perform a systematic review of papers that address the impact of irrelevant information, in order to identify theories that can explain the phenomenon and suggest techniques that can reduce the impact.

In addition, I consider to replicate and extend the findings presented in the thesis, in the following respects:

Replication of the studies presented in the thesis with:

- Larger samples. This is especially important for the study reported in Paper II, because this study was conducted with a very low number of subjects.
- Samples from other populations. The studies reported in Papers I, II and III were conducted with experienced software developers as subjects. Conducting similar experiments with samples from other populations, such as inexperienced software developers and project leaders, would increase our knowledge.
- Other estimation tasks. The tasks that the subjects estimated in the studies reported in Papers I and II were relatively small and most of the subjects had experience with the relevant technologies. Investigating the impacts on other types of tasks would be useful for our understanding.
- More realistic environment. It would be interesting to replicate the studies reported in Papers I and II at the subjects' work-place, e.g., in a setting where they have access to estimation guidelines, historical data, and, other experts.
- Reduced level of anonymity. Replicating the survey without project and company anonymity could offer some insight into the little-studied topic of the effect of anonymity in software engineering surveys, e.g., would respondents answer differently if they, as they do in most field settings, have to describe explicitly the clients and projects they used as background for their opinions.
- Assessment of the review questions presented in Papers IV and V by external researchers. At present, there is a risk of research bias, because the authors assessed the review questions themselves.

Studies that investigate other aspects of the phenomena investigated in this thesis:

- Experiments that investigate the actual effect of the client related estimation error factors reported in Paper III.
- Experiments that investigate irrelevant information and inconsistency and where the tasks, as opposed to the studies described in Papers I and II, are actually implemented.
- Experiments that investigate the potential of reduced inconsistency and impact of irrelevant information by training estimators and using especially tailored guidelines for estimating.
- Systematic reviews of related research results from other fields. Topics related to the prediction of events have been investigated for decades, and sometimes centuries, in fields such as economics and psychology. An extensive summary of their findings may lead to improved processes of effort estimation in software development contexts.

7. Concluding Remarks

The papers included in the thesis address a variety of topics related to effort estimation, e.g. topics as different as inconsistency in cognitive estimation processes and the use of terminology. A stronger focus in the thesis would have enabled an in-depth overall contribution to a more specific research question than "How can expert-judgment based effort estimation processes be changed to reduce software effort estimation error in software development projects?". One of the strengths of the spread of the research topics of this thesis is that it points to the variety of topics that need to be address by estimation research to reduce effort estimation error. In addition, the variety of topics enables the use of a variety of research methods and is, consequently, important in my researcher education.

References

[1] B. Anda, E. Angelvik, and K. Ribu, "Improving estimation practices by applying use case models," *proc. Product Focused Software Process Improvement. 4th International Conference*, pp. 383-397, 2002.

[2] J. Aranda and S. Easterbrook, "Anchoring and Adjustment in Software Estimation," *proc. European software engineering conference*, pp. 346-355, 2005.

[3] B. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, *Software cost estimation with Cocomo II*. New Jersey: Prentice-Hall, 2000.

[4] B. W. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering*, vol. 10, no. 1, pp. 4-21, 1984.

[5] L. C. Briand and I. Wieczorek, "Resource estimation in software engineering," in *Encyclopedia of software engineering*, J. J. Marcinak, Ed., 2nd ed. New York: John Wiley & Sons, 2002, pp. 1160-1196.

[6] M. Cohn, *Agile Estimating and Planning*: Prentice Hall PTR, 2005.

[7] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*. Menlo Park, California: Benjamin Cummings, 1986.

[8] J. S. Edwards and T. T. Moores, "A conflict between the use of estimating and planning tools in the management of information systems," *European Journal of Information Systems*, vol. 3, no. 2, pp. 139-147, 1994.

[9] L. Farr and H. J. Zagorski, "Quantitative analysis of programming cost factors: A progress report," *proc. ICC Symposium Proc. Economics of Automatic Data Processing*, pp. 167-180, 1965.

[10] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, vol. 29, no. 11, pp. 985-995, 2003.

[11] F. J. Heemstra and R. J. Kusters, "Function point analysis: Evaluation of a software cost estimation model," *European Journal of Information Systems*, vol. 1, no. 4, pp. 223-237, 1991.

[12] J. Hihn and H. Habib-Agahi, "Cost estimation of software intensive projects: A survey of current practices," *proc. International Conference on Software Engineering*, pp. 276-287, 1991.

[13] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679-688, 2006.

[14] M. Jørgensen, "How much does a vacation cost?," *Software Engineering Notes*, vol. 28, no. 6, p. 30, 2003.

[15] M. Jørgensen, "How much does a vacation cost? or What is a software cost estimate?," *Software Engineering Notes*, vol. 28, no. 6, pp. 5-5, 2003.

[16] M. Jørgensen, "The "Magic Step" of Judgment-Based Software Effort Estimation," *proc. International Conference on Cognitive Economics*, 2005.

[17] M. Jørgensen, " Estimation of Software Development Work Effort:Evidence on Expert Judgment and Formal Models," *forthcoming in International Journal of Forecasting*, 2006.

[18] M. Jørgensen and K. Moløkken-Østvold, "Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 993-1007, 2004.

[19] M. Jørgensen and M. Shepperd, "A systematic Review of Software Development Cost Estimation Studies," *Forthcoming in IEEE Transactions on Software Engineering*, 2005.

[20] M. Jørgensen and K. Moløkken-Østvold, "How Large Are Software Cost Overruns? Critical Comments on the Standish Group's CHAOS Reports," *Information and Software Technology*, vol. 48, no. 4, pp. 522-528, 2006.

[21] B. Kitchenham, *Software Metrics: Measurement for Software Process Improvement*: Blackwell Publishers, 1996.

[22] B. Kitchenham, "Procedures for Performing Systematic Reviews," Keele University, Keele, Technical Report 2004.

[23] B. A. Kitchenham, L. M. Pickard, S. G. MacDonnel, and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proceedings Software*, vol. 148, no. 3, pp. 81-85, 2001.

[24] R. J. Kusters, M. J. I. M. Genuchten, and F. J. Heemstra, "Are software cost-estimation models accurate?," *Information and Software Technology*, vol. 32, no. 3, pp. 187-190, 1990.

[25] C. Larman, *Applying UML and Patterns: an introduction to object-oriented analysis and design and iterative development*, 3rd ed. Upper Saddle River: Pearson Education, Inc., 2005.

[26] A. L. Lederer and J. Prasad, "Causes of Inaccurate Software Development Cost Estimates," *Journal of Systems and Software*, vol. 31, pp. 125-134, 1995.

[27] J. Li and G. Ruhe, "A comparative study of attribute weighting heuristics for effort estimation by analogy," *proc. Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering*, pp. 66-74, 2006.

[28] C. Lokan and E. Mendes, "Cross-company and Single-company Effort Models using the ISBSG Database: a Further Replicated Study," *proc. Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering*, pp. 75-84, 2006.

[29] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting Methods and Applications*, 3rd ed. New York: John Wiley & Sons, Inc., 1998.

[30] P. E. Meehl, "When shall we use our heads instead of the formula?," *Journal of Counseling Psychology*, vol. 4, no. 4, pp. 268-273, 1957.

[31] E. Mendes and N. Mosley, "Comparing effort prediction models for Web design and authoring using boxplots," *proc. Australasian Computer Science Week*, pp. 125-133, 2001.

[32] Y. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada, "Method to estimate parameter values in software prediction models," *Information and Software Technology*, vol. 33, no. 3, pp. 239-243, 1991.

[33] K. Moløkken and M. Jørgensen, "A review of software surveys on software effort estimation," *proc. International Symposium on Empirical Software Engineering*, pp. 223-230, 2003.

[34] K. Moløkken-Østvold, M. Jørgensen, S. Tanilkan, H. Gallis, A. Lien, and S. Hove, "A Survey on Software Estimation in the Norwegian Industry," *proc. Metrics '04*, pp. 208-219, 2004.

[35] I. Myrtveit, M. Shepperd, and E. Stensrud, "Reliability and Validity in Comparative Studies of Software Prediction Models," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380-391, 2005.

[36] S. Oskamp, "Overconfidence in case study judgments," *Journal of Consulting Psychology*, vol. 29, pp. 261-265, 1965.

[37] J. Paynter, "Project estimation using screenflow engineering," *proc. International Conference on Software Engineering: Education and Practice*, pp. 150-159, 1996.

[38] J. D. Procaccino, J. M. Verner, S. P. Overmyer, and M. E. Darter, "Case Study: factors for early prediction of software development success," *Information and Software Technology*, vol. 44, pp. 53-62, 2002.

[39] J. Ropponen and K. Lyytinen, "Components of Software Development Risk: How to Address them? A project Manager Survey," *IEEE Transactions on Software Engineering*, vol. 26, no. 2, 2000.

[40] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit, "An empirical validation of the relationship between the magnitude of relative error and project size," *proc. International Software Metrics Symposium*, 2002.

[41] T. R. Stewart, "Improving Reliability of Judgmental Forecasts," in *Principles of Forecasting*, J. S. Armstrong, Ed. Boston: Kluwer Academic Publishers, 2001, pp. 81-106.

[42] R. C. Tausworthe, "The work breakdown structure in software project management," *Journal of Systems and Software*, vol. 1, no. 3, pp. 181-186, 1980.

[43] J. M. Verner, S. P. Overmyer, and K. W. McCain, "In the 25 years since The Mythical Man-Month what have we learned about project management?," *Information and Software Technology*, vol. 41, pp. 1021-1026, 1999.

[44] G. M. Weinberg and E. L. Schulman, "Goals and performance in computer programming," *Human Factors*, vol. 16, no. 1, pp. 70 - 77, 1974.

[45] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*: Kluwer Academic Publishers, 2000.

[46] H. Zukier, "The dilution effect: The role of correlation and the dispersion of predictor variables in the use of diagnostic information," *Journal of Personality and Social Psychology*, vol. 43, pp. 1164-1174, 1982.

Paper I:

The Impact of Irrelevant Information on Estimates of Software Development Effort

Stein Grimstad[1,2] and Magne Jørgensen[2]

[1]*Simula Research Laboratory*

[2]*University of Oslo*

Abstract

Software professionals typically estimate software development effort based on a requirement specification. Parts of this specification frequently contain information that is irrelevant to the estimation of the actual effort involved in the development of software. We hypothesize that effort-irrelevant information sometimes has a strong impact on effort estimates. To test this hypothesis, we conducted two controlled experiments with software professionals. In each of the experiments, the software professionals received specifications describing the same requirements. However, we gave one group of the software professionals a version of the requirement specification where we had included additional, effort-irrelevant, information. In both experiments we observed that the estimates of most likely effort increased when the estimates were based on requirement specifications that contained the information irrelevant to development effort. The results suggest that when estimation-irrelevant information is included as input to expert judgment-based estimation processes, the estimators find it difficult to distinguish between the estimation-relevant and the estimation-irrelevant information. A possible consequence of our findings is that estimation-irrelevant information should be removed from the requirement specification prior to the use of it as input to estimation work.

1. Introduction

Software projects frequently overrun their effort estimates [1]. This is a major concern for the software industry, because the quality of software effort estimates directly affects companies' ability to compete. Poor estimation performance often causes budget overruns, delays, lost contracts and low-quality software.

A recent review [2] summarizes findings suggesting that expert judgment-based estimation is the most popular estimation method in the software industry. Typically, studies report that 70-80% of industrial estimates are made by experts without using formal estimation models. The review summarizes studies of expert judgment- and model-based effort estimates and concludes that the evidence does not support a replacement of expert judgment with estimation models. Although there are studies that have identified factors that affect the judgment-based effort estimates [2, 3], our understanding of the steps and biases involved in expert estimation is limited [4]. The popularity of the method, and the lack of knowledge about it, indicates that a better understanding of expert estimation may be required to meet the software industry's demand for more accurate effort estimates.

There are many factors that are relevant to the effort of software development [5, 6], e.g., amount of functionality, focus on cost control in the project and implementation technology. In an ideal world, we would like the estimate to be based on only relevant factors and not be affected by information that has no relation to the actual effort. Information about the choice of GUI colors in a web system should, for example, not affect the estimate of the effort required to develop a new order engine. Neither should the font size and margins of a requirement specification affect the estimate. However, an unpublished experiment conducted by the second author of this paper on computer science students found that this could be the case! In that experiment, half of the students estimated development effort based on a short requirement specification, and the other half estimated based on a long specification. The text in the two specifications was identical, but line-spacing, page set up and font size were adjusted so that the long version of the specification was seven pages and the short version only one page long. The students exposed to the long version provided on average 16% higher effort estimates. This effect caused by irrelevant information is consistent with research in other fields [7-11]. Hristova et al. [10], for example, report that the colour of the text influenced price judgments, and, Gaeth and Shanteau [11] report that experienced soil judges are influenced by irrelevant factors in soil judgment.

Software effort estimates are often based on requirement specifications where the information varies in precision, structure and relevance. It is rare that all the information in a requirement specification is relevant for estimating software development effort. Estimation-irrelevant information is included in requirement specifications for a number of reasons, such as the following: insufficient time is spent on removing information of less relevance (e.g., the text is copied from a previous specification), the author lacks knowledge of what to include in a requirement specification, the information is useful for purposes other than software effort estimation. This study investigates empirically whether the presence of information that is irrelevant to estimation of software effort affects software professionals' effort estimates. The research question is as follows:

RQ: Are software professionals' estimates of most likely effort affected by estimation-irrelevant information in the requirement specifications?

By the term "estimation-irrelevant information" we mean information that does not have a direct or indirect *casual* relationship to software development cost. Notice that, as we interpret it in this paper, information can be estimation-irrelevant even if it has a *correlation* to actual effort. (Correlations may indicate a causal relationship at a deeper level, but do not themselves constitute a causal relationship.) For example, the length of the requirement specification may correlate with development effort. We would categorize the length of the requirement specification as effort estimation *relevant* if a difference in length is caused by differences in the amount of development effort demanding requirements. Length or the requirement specification would, on the other hand, be categorized as *irrelevant* information if a difference in length is caused by differences in text formatting or by inclusion of information that has nothing to do with the development of the software.

Our hypothesis was tested in two controlled experiments. In both experiments, half of the software professionals estimated a software development task based on a requirement specification where we had introduced estimation-irrelevant information, and the other half estimated effort based on the same specification, with the estimation-irrelevant information removed.

The remainder of the paper is organized as follows. Section 2 describes related work on the effect of irrelevant information. The experiments are presented in Sections 3 and 4. Section 5 discusses the results. Section 6 summarizes the paper and provides recommendations.

2. Related work

In order to find software cost estimation studies related to our research question, we searched the BESTweb database[*] for studies that investigate empirically the impact of irrelevant information on software cost estimates. BESTweb is an online library of estimation papers that claims to include nearly all journal papers and many of the conference papers on software cost estimation. The selection of papers included in BESTweb are described in [12]. At the time of the review, the BESTweb library contained 964 estimation-relevant articles. We also included a recent study that we are aware of, not included in the BESTweb database. The following studies on the impact from irrelevant information in software engineering contexts were identified:

- Jørgensen and Sjøberg [13] report that preplanning effort estimates can have a major impact on detailed planning effort estimates, even when the estimators are told that the early estimates are not based on historical data or expert knowledge, i.e., should not be considered as relevant information to the estimation process. The estimators' awareness of the impact of the irrelevant information was low.
- Jørgensen and Sjøberg [14] report that information about the customer's expectations can significantly affect most-likely estimates of software development, even when the subjects are explicitly told that the customer's expectation is not an indicator of the actual effort. The estimators did not notice this effect or assessed it to be low.
- Aranda and Easterbrook [15] report that including customer expectations of cost, which are clearly marked as irrelevant in the requirement specification, can have a large impact on cost estimates. The impact could not be explained by the subjects' estimation experience.

In these studies, the irrelevant information is presented to the subjects as some sort of initial estimate that is irrelevant to the subjects' estimation process. This special case of irrelevant information is typically termed "anchoring". The impact of anchors is strong, and has been demonstrated in many domains [16]. However, the estimates may, as reported by research in other fields [7, 17-19], be affected severely by other types of irrelevant information. These studies have shown that the introduction of irrelevant information can lead to increased estimation error, reduced estimation reliability, less learning, and increased over-confidence in one’s own estimates. There is also

[*] available at http:///www.simula.no/BESTweb

evidence that personal characteristics, such as domain expertise [20], attention ability [21] and handedness [22], can make an estimator more likely to be subject to the effects of irrelevant information.

Results from other research fields should be considered with some care, because most of the studies are conducted in contexts that differ from software effort estimation. Hence, we need to carry out studies in contexts similar to those met by software professionals estimating development effort.

Consequently, the main contributions of this paper are these: 1) to study the effect of irrelevant information on judgment in a software development effort context, and 2) to investigate empirically the impact of textual estimation irrelevant information, i.e., the effect of non-numerical irrelevant information on effort estimates. Textual irrelevant information may be more common in software development effort estimation contexts than irrelevant numerical anchors, but we have been unable to find previous studies on this topic. We have found no other study on the impact of textual irrelevant information in software development effort estimation contexts.

3. Experiment 1

Experiment 1 was designed to test two issues: 1) the impact of irrelevant information on the effort estimates, and 2) the impact of level of specification precision on the estimates. This paper focuses on only the first issue, i.e., the impact of estimation-irrelevant information. For this reason, the impact of the level of precision will only be discussed related to the possibility of interaction effects between precision and irrelevant information. Section 3.1 describes and discusses the design of the experiment, while Section 3.2 presents the results.

3.1 Design of Experiment

We wanted to investigate our research question in a realistic setting, i.e., software professionals completing estimation tasks similar to those they normally complete. In order to isolate the factors we wished to study, we applied a 2x2 factorial design with random allocation of treatment. The two binary factors were related to the presence of irrelevant information and the degree of precision of the requirement specification.

Participants

The experiment was conducted at a conference for software developers (JavaZone 2005[*]). There were 76 software professionals participating in the experiment. On average, the participants each had 10 years of experience as software developer. The sample is self-selected in the sense that the participants were those who chose to attend a particular lecture on software cost estimation. This suggests that the participants might be more than averagely interested in estimation. This, in turn, may imply that any biases in the sample of participants are likely to be in the direction of better than average estimation expertise.

The randomized allocation of treatment is likely to have eliminated systematic differences in personal characteristics within the sample.

Estimation task

Cooksey [23] cautions that experts are especially sensitive to the realism and familiarity in judgmental tasks. The estimation task was therefore based on an actual industrial task to ensure realism. To increase the likelihood of familiarity, we chose a small development task. This is based on the belief that software developers more frequently estimate smaller tasks, i.e. parts of a project, than the entire project, i.e., the likelihood of previous experience with similar estimation tasks increases with small tasks. In addition, the estimation of a small task would increase the similarity of time used for estimation in the experiment and in a real-world context. The participants estimated the effort required to write a simple program that retrieves a file from a remote server, validates the data and stores the data in an existing database. We believe that this is a rather general task that requires little specialized technology and domain knowledge.

We created four variants of the same requirement specification: i) a high-level requirement specification without irrelevant information being introduced, ii) a high-level requirement specification that included irrelevant information, iii) a detailed requirement specification without irrelevant information being introduced, and iv) a detailed requirement specification that included irrelevant information. The difference between the high-level and the detailed requirement specification was that the detailed requirement specification included explicit validation rules and a

[*] see http://www.javazone.no

complete example of file format and file policy. The irrelevant information consisted of information about end users' work processes, a description of the selection criteria that were used for selection of data providers, and information about systems that their implementation would not have to integrate with i.e., information that should not lead to more or less development effort. The requirement specifications are shown in Appendix A.

Two experienced developers validated the requirement specification. These two developers were asked to evaluate whether the four variants of the specification: 1) lacked any information normally found in this type of specification, 2) contained any errors, 3) described a realistic development task, and 4) was representative of tasks normally implemented by the conference attendees likely to participate in our experiment. In addition, they were requested to evaluate whether the irrelevant information that was introduced really was irrelevant for the purpose of estimating the software development effort.

Treatment

The participants were divided into four groups (groups A, B, C and D) with different variants of the requirement specification allocated to each group; see Table 1.

Table 1 Treatment in Experiment 1

Group	Detailed requirements	Irrelevant information
A	No	No
B	Yes	No
C	No	Yes
D	Yes	Yes

The specifications were handed out so that every fourth participant, by physical location, was allocated to the same group. The estimation task was included in a set of three other experiments, and a survey.

The participants took about 10 minutes to complete the effort estimation task, which is not unrealistic for this type of small estimation task. The participants' responses were collected immediately after the allocated time had expired.

Measurement

The participants were asked to estimate the most likely effort (in hours) they would need to implement the specified program. To measure the participants' confidence in their most likely estimate, we asked them to provide a minimum-maximum interval (in hours) that they were 90% certain would contain the actual effort. In the results, we present the relative width of the minimum-maximum interval as a measure of confidence. The relative width of the minimum-maximum intervals is calculated by the following formula:

RWidth = (Maximum value – Minimum value) / Estimate of most likely effort

The lower the RWidth, the higher the confidence in the accuracy of the estimate.

3.2 Results

An analysis of potential outliers revealed one obvious outlier that was removed from the data set. This participant submitted an effort estimate that was very much higher than that of the other participants, i.e., he estimated the effort to be 1250 work-hours while the mean value of the remaining participants was 29.6 work-hours. We believe that this very high estimate indicates that the participant either did not take the task seriously, did not have the skill required to estimate meaningfully, or estimated something other than development and unit testing of the program specified in the requirement specification.

The results are displayed in Table 2 (estimates of most likely effort) and Table 3 (relative width of the minimum-maximum intervals).

Table 2 Estimates of Most Likely Effort (work-hours)

Group	N	Mean	Median	Min	Max	Stdv
A (high-level specification with no irrelevant info)	19	17.2	11.0	4	60	14.2
B (detailed specification with no irrelevant info)	18	22.2	17.5	5	70	17.5
C (high-level specification with irrelevant info)	20	32.8	30.0	8	80	20.1
D (detailed specification with irrelevant info)	18	46.7	24.5	4	250	65.5
A + B (no irrelevant information)	37	19.7	15.0	4	70	15.9
C + D (irrelevant information)	38	39.3	27.5	4	250	47.2

Table 3 Relative Width of Minimum-Maximum Interval (RWidth)

Group	N	Mean	Median	Min	Max	Stdv
A (high-level specification with no irrelevant info)	19	1.44	1.20	0.38	3.82	0.97
B (detailed specification with no irrelevant info)	18	1.27	0.83	0.22	6.00	1.32
C (high-level specification with irrelevant info)	20	0.86	0.78	0.33	1.50	0.40
D (detailed specification with irrelevant info)	18	0.99	0.85	0.47	2.00	0.44
A + B (no irrelevant information)	37	1.36	1.00	0.22	6.00	1.14
C + D (irrelevant information)	38	0.92	0.82	0.33	2.00	0.42

The results show that the participants that received requirement specifications with irrelevant information submitted, on average, higher effort estimates than the participants that did not receive irrelevant information (mean of 19.7 vs. 39.3 work-hours). To analyze the effect of the independent variables "Irrelevant information introduced" (yes/no), "Detailed specification" (yes/no) and the interaction between the two on the dependent variable "Estimates" we fitted a General Linear Model (GLM). A log-transformation of the dependent variable was required to achieve a normal distribution of the residuals. The analysis shows that the impact of estimation-irrelevant information is highly significant (p=0.01), and that the interaction effect between the binary variables (related to presence of irrelevant information and/or level of specification) is not significant (p=0.38). The relative effect size of adding irrelevant information (based on Least Square Means estimates of irrelevant/relevant information) is +72%.

Surprisingly, the participants' confidence in the accuracy of their own estimates increased, i.e., the relative minimum-maximum interval width decreases, when irrelevant information was added! The mean relative width was 1.36 without irrelevant information, yet 0.92 when irrelevant information was included. The difference was statistically significant (p=0.03) applying log(relative width) to achieve a normal distribution of the residuals and GLM to compensate for any interaction effects between the independent variables related to irrelevant information and level of specification. This means that although the inclusion of irrelevant information affected the estimate, and consequently affected the level of realism negatively, confidence in the accuracy of the effort estimates actually increased.

4. Experiment 2

The second experiment was designed to: 1) test the robustness of the results in Experiment 1 on a different estimation task, 2) to further investigate the impact of irrelevant information, and 3) to investigate the effect of asking participants to explain the basis of their estimates, i.e., a weak variant of justification of effort estimates. As before, our focus is on the impact of irrelevant information. We will, therefore, only discuss the impact of justification in relation to possible interaction effect with level of irrelevant information. The design is discussed in Section 4.1. The results are presented in Section 4.2

4.1 Design of Experiment

The design of Experiment 2 was a 2x2 factorial design similar to the design of Experiment 1. The binary factors were presence/no presence of irrelevant information and justification/no justification of the estimates.

Participants

The experiment was conducted at an estimation seminar for professional software developers. We did not collect information about the participants' background in this experiment, but it is probable that the participants in Experiments 1 and 2 were similar with respect to experience and organizational role. This belief is based on the distribution of invited companies, and discussions with seminar attendees before and after the seminar.

Estimation Task

As in Experiment 1, we tried to create an estimation task that was small enough to be estimated realistically in a short experiment, representative for real-world estimation tasks, and based on assumptions about the use of technologies well known to the participants. The task was based on the estimation of a simple web application that registered seminar attendees in a database.

In this experiment, the treatments were as follows: requirements specification i) without irrelevant information and no request for justification of the estimate, ii) with irrelevant information and no justification, iii) without irrelevant information, but with justification, and, iv) with irrelevant information and justification. Justification of estimates was obtained by asking the participants to assess how relevant different parts of the requirement specification were for their estimates. The irrelevant information consisted of a description of a complex system that would, sometime in the future, replace the program they estimated. If this information had any relevance at all for the development effort, the impact should, we think, have been that information about future replacement would reduce the actual effort due to the lesser importance placed on long-term quality issues such as maintainability. The requirement specifications are shown in Appendix B. The different versions of the requirement specifications were, similarly to Experiment 1, validated with respect to quality, realism and whether the introduced additional information really was irrelevant for the purpose of effort estimation by software professionals.

Treatment

The participants were divided into four groups (group A, B, C and D) with different treatments allocated to each group, see Table 4.

Table 4 Treatment in Experiment 1

Group	Irrelevant information	Justification
A	No	No
B	Yes	No
C	No	Yes
D	Yes	Yes

As in Experiment 1, the participants were randomly allocated to treatment (by physical location in the seminar room), and the tasks were completed in a time frame and conditions similar to those in Experiment 1, i.e., the time spent on the estimate was about 10 minutes. The experimental task was included in a set of two experiments and one brief survey.

Measurement

As in Experiment 1, the participants were asked to estimate the most likely effort in work-hours and to provide a minimum-maximum interval (also in work-hours) that they were 90% certain would contain the actual effort.

4.2 Results

One participant in Group a (200 work-hours) and three participants in Group C (estimates of 150, 300 and 400 work-hours) were considered to be outliers and removed. These participants were removed on the basis that the very high effort estimates made it likely that they either did not take the task seriously, did not have sufficient expertise, or misunderstood the task.

The results are displayed in Table 5 (estimates of most likely effort), and in Table 6 (relative width of the minimum-maximum intervals).

Table 5 Estimates of Most Likely Effort

Group	N	Mean	Median	Min	Max	Stdv
A (basic)	21	11.8	8	0.5	40	11.8
B (irrelevant information)	23	14.8	8	1.0	40	12.8
C (justification)	20	20.5	8	0.5	120	30.0
D (irrelevant information and justification)	24	22.5	11	3.0	100	24.4
A + C (no irrelevant information)	41	16.0	8	0.5	120	22.7
B + D (irrelevant information)	47	18.7	10	1.0	100	19.8

Table 6 Relative Width of Minimum-Maximum Interval (RWidth)

Group	N	Mean	Median	Min	Max	Stdv
A (basic)	21	1.22	1.00	0.40	3.80	0.73
B (irrelevant information)	23	1.21	1.20	0.38	2.10	0.55
C (justification)	20	1.18	1.00	0.25	2.67	0.68
D (irrelevant information and justification)	24	1.24	1.26	0.40	3.00	0.63
A + C (no irrelevant information)	41	1.20	1.00	0.25	3.80	0.70
B + D (irrelevant information)	47	1.22	1.20	0.38	3.00	0.58

The results show that (i) the participants who received requirement specifications with irrelevant information submitted higher effort estimates than those that did not receive irrelevant information (mean of 16.0 vs. 18.7 work-hours), and that (ii) the impact of irrelevant information seemed to be somehow moderated when the participants had to justify their estimates (mean of 11.8 vs. 14.8 work-hours when they did not have to justify their estimate, mean of 20.5 vs. 22.5 when they did have to provide justification). Statistical analysis of the data, similar to that in Experiment 1, i.e., GLM analysis of log(ML estimates) and the binary variables “Irrelevant information” (yes/no) and “Justification” (yes/no), shows that the impact of estimation-irrelevant information is significant (p=0.08). The interaction effect (irrelevant information/justification) on the estimates is not significant (p=0.83). The relative effect size of adding irrelevant information (based on Least Square Means estimates of irrelevant/relevant information) is +52%. The results strengthen the results from Experiment 1 as they clearly point in the same direction.

The participants’ confidence in their own estimates, measured as mean relative width of minimum-maximum intervals, was not much affected by irrelevant information in this experiment with mean values of 1.22 (irrelevant information included) vs. 1.20 (without irrelevant information). Statistical analysis, by GLM analysis of log (relative width), of statistical significance of difference confirms this (p=0,74).

5. Discussion

In both experiments, the average estimate of most likely effort increased when estimation irrelevant information was included. The main reason for this effect of irrelevant information in our experiments may be the use of simple, unconscious estimation processes, so-called judgmental heuristics by the participants. Such heuristics are frequently used to solve complex problems, where the human mind is not capable of implementing the “normatively correct” processes [24].

To keep the estimation processes simple, the participants in our study may have based their effort estimates on easily available variables with no causal relationship to effort, on the assumption that these variables usually *correlate* well with amount of effort. The length of the text, the number of systems mentioned, or simply the assumption that everything in the requirement specification is relevant [9] may be examples of irrelevant information used as indicator of effort. Judgmental heuristics, such as those used in software effort estimation, are often unconscious processes (often they have originally been analytic and evolved into tacit processes as they have been used repeatedly with success [24]). The unconscious use of variables means that the effort estimates may have been affected by information elements that the estimators, when asked about it, would admit are irrelevant in the current estimation situation.

Two elements of such, more or less, unconscious heuristics are “estimation-by-analogy” and “first impression”:

- *Estimation-by-analogy*: Estimation by analogy is quite common in software effort estimation [25]. Estimation by analogy is based, to some extent unconsciously, on retrieving one or more tasks from memory (the analogies) that resemble the task that is going to be estimated, and then creating the estimate based on properties and actual effort of the retrieved tasks. In the experiments, the selection of analogies might have been based on surface cues (e.g., the number of systems mentioned, the technical platform information) or in-depth cues (e.g., the steps involved in solving the task). Irrelevant information might have surface similarity to previous tasks that differ in the underlying structure. This means that the irrelevant information might have led to misleading, or at least other, analogies compared to the situation without irrelevant information.
- *"First impression"*: People can be strongly affected by their first impression when making predictions and other decisions under uncertainty [26], e.g. studies have found that court decisions are affected severely by the jury's first impression. The estimation process might be based on an early, unconscious, categorization ("first impression") of the estimation task into a predefined category. When the estimate is created, it is strongly influenced by the initially chosen category, e.g., that the task looks like a "medium-large task". In both experiments, the irrelevant information was placed early in the requirement specification, and might therefore have caused an incorrect "first impression" that was difficult to change with more information, e.g., by reading further in the requirement specification, as confirming evidence has a stronger effect than evidence that does not fit with the initially chosen task category, i.e., the effect of "theory-loaded observations" [27, 28].

More studies on the effect of the amount, the type, the extremity, the framing and the placement of irrelevant information are needed to better understand when and how it affects effort estimates. Until we know how to neutralize the effect of irrelevant information, we believe the best strategy is to try to avoid it altogether, particularly in the early stages of the estimation process.

Some might find this advice counterintuitive, because average effort estimates increased in both experiments when irrelevant information was included, and it is well known that software cost estimates are usually too optimistic. However, other irrelevant information can cause effort estimates to decrease. Unless the estimation process is based on information that is relevant for the actual use of effort, systematic improvement of judgment-based effort estimation may be very difficult.

6. Summary and Recommendations

It may be the exception, rather than the rule, that all information in a requirement specification is relevant for estimation of software development effort. Does this not matter or is it essential to avoid irrelevant information in requirement specifications? We designed two experiments to answer the research question of whether effort estimates were affected by the presence of irrelevant information.

The two experiments (with 76 and 92 participants) answered this research question with the observation that estimation-irrelevant information in requirement specifications *strongly* affected the software effort estimates. The average effort estimates increased significantly in both experiments when estimation-irrelevant information was included. In addition, the results of Experiment 1 suggest that the estimators may also become more confident in the accuracy of their own estimates when they are exposed to irrelevant information.

The magnitude of the effect differed and we currently have a quite incomplete understanding of how, when and how much different irrelevant information affects cost estimation. Consequently, further research is needed.

Until we have a better understanding of the impact of irrelevant information on expert judgment-based effort estimates, we believe it to be essential that irrelevant information is *removed* from requirement specifications before presented to the estimators [29]. If this is impossible, it may be a good idea to highlight and present early the most relevant information to avoid incorrect first impressions [30]. The removal of irrelevant information is important even when using formal estimation models, i.e., formal estimation models are typically based on expert judgment-based input. This input may also be affected by irrelevant information.

Acknowledgment

Thanks to Dr. Erik Arisholm at Simula Research Laboratory for criticism and suggestions that improved the study design and the statistical analysis. Also, thanks to the Norwegian Research Council's SPIKE project for funding.

References

1. Moløkken, K. and M. Jørgensen. *A review of software surveys on software effort estimation*. in *International Symposium on Empirical Software Engineering*. 2003. Rome, Italy: Simula Res. Lab. Lysaker Norway.
2. Jørgensen, M., *A review of studies on expert estimation of software development effort.* Journal of Systems and Software, 2004. **70**(1-2): p. 37-60.
3. Armstrong, J.S., ed. *Principles of forecasting: A handbook for researchers and practitioners*. International Series in Operations Research & Management Science. 2001, Kluwer Academic Publishers: Boston. XII, 849 s. : ill.
4. Jørgensen, M. *The "Magic Step" of Judgment-Based Software Effort Estimation*. in *International Conference on Cognitive Economics*. 2005. Sofia, Bulgaria: NBU Press.
5. Boehm, B., et al., *Software cost estimation with Cocomo II*. 2000, New Jersey: Prentice-Hall.
6. Albrecht, A.J. and J.E. Gaffney Jr, *Software function, source lines of code, and development effort prediction: A software science validation.* IEEE Transactions on Software Engineering, 1983. **9**(6): p. 639-648.
7. Whitecotton, S.M., D.E. Sanders, and K.B. Norris, *Improving predictive accuracy with a combination of human intuition and mechanical decision aids.* Organizational Behaviour and Human Decision Processes, 1998. **76**(3): p. 325-348.
8. Lim, J.S. and M. O'Connor, *Judgmental forecasting with time series and causal information.* International Journal of Forecasting, 1996. **12**(1): p. 139-153.
9. Tversky, A. and D. Kahneman, *Judgment under uncertainty: Heuristics and biases.* Science, 1974. **185**: p. 1124-1131.
10. Hristova, P., G. Petkov, and B. Kokinov. *The Influence of Irrelevant Information on Price Judgment*. in *Advances in Cognitive Economics*. 2005. Sofia, Bulgaria: NBU Press.
11. Gaeth, G.J. and J. Shanteau, *Reducing the Influence of Irrelevant Information on Experienced Decision Makers*, in *Judgment and Decision Making: An interdisciplinary reader*, T. Connoly, H. Arkes, and K.R. Hammond, Editors. 2003, Cambridge University Press: Cambridge. p. 305-321.
12. Jørgensen, M. and M. Shepperd, *A systematic Review of Software Development Cost Estimation Studies.* Submitted to IEEE Transactions on Software Engineering, 2005.
13. Jørgensen, M. and D.I.K. Sjøberg, *Impact of effort estimates on software project work.* Information and Software Technology, 2001. **43**(15): p. 939-948.
14. Jørgensen, M. and D.I.K. Sjøberg, *The impact of customer expectation on software development effort estimates.* International Journal of Project Management, 2004. **22**: p. 317-325.
15. Aranda, J. and S. Easterbrook. *Anchoring and Adjustment in Software Estimation*. in *European software engineering conference*. 2005. Lisbon, Portugal: ACM Press.
16. Epley, N., *A Tale of Tuned Decks? Anchoring as Accessibility and Anchoring as Adjustment*, in *Blackwell Handbook of Judgment and Decision Making*, D.J. Koehler and N. Harvey, Editors. 2004, Blackwell Publishing Ltd. p. 240-257.
17. Zukier, H., *The dilution effect: The role of correlation and the dispersion of predictor variables in the use of diagnostic information.* Journal of Personality and Social Psychology, 1982. **43**: p. 1164-1174.
18. Stewart, T.R., *Improving Reliability of Judgmental Forecasts*, in *Principles of Forecasting*, J.S. Armstrong, Editor. 2001, Kluwer Academic Publishers: Boston. p. 81-106.
19. Oskamp, S., *Overconfidence in case-study judgments.* Journal of Consulting Psychology, 1965. **29**(3): p. 261 - 265.
20. Ettenson, R., J. Shanteau, and J. Krogstad, *Expert judgment: Is more information better.* Psychological Reports, 1987. **60**(1): p. 227-238.
21. Marzocchi, G., et al., *The disturbing effect of irrelevant information on arithmetic problem solving in inattentive children.* Developmental Neuropsychology, 2002. **21**(1): p. 73-92.

22. Jasper, J. and S. Christman, *A neuropsychological dimension for anchoring effects.* Journal of Behavioral Decision Making, 2005. **18**(5): p. 343.

23. Cooksey, R.W., *Judgment Analysis: Theory, Methods and Applications.* 1996, London: Academic Press, Inc.

24. Gigerenzer, G., *Fast and Frugal Heuristics: The tools of Bounded Rationality*, in *Blackwell Handbook of Judgment and Decision Making*, D.J. Koehler and N. Harvey, Editors. 2004, Blackwell Publishing: Oxford, UK. p. 63-88.

25. Heemstra, F.J., *Software cost estimation.* Information and Software Technology, 1992. **34**(10): p. 627-639.

26. Keren, G. and K.H. Teigen, *Yet Another Look at the Heuristics and Biases Approach*, in *Blackwell Handbook of Judgment and Decision Making*, D.J. Koehler and N. Harvey, Editors. 2004, Blackwell Publishing: Oxford, UK. p. 89-109.

27. Brehmer, B., *In one word: Not from experience.* Acta Psychologica, 1980. **45**: p. 223-241.

28. Einhorn, H.J. and R.M. Hogarth, *Confidence in judgment: Persistence of the illusion of validity.* Psychological Review, 1978. **85**(5): p. 395-416.

29. Williams, C., *The effect of an irrelevant information dimension on "same-different" judgments of multi-dimensional stimuli.* Quarterly Journal of Experimental Psychology, 1974. **26**: p. 26-31.

30. Conway, J.M., R.A. Jako, and D.F. Goodman, *A meta-analysis of interrater and internal consistency reliability of selection interviews.* Journal of Applied Psychology, 1995. **80**: p. 565-579.

Paper II:

Inconsistency of Expert Judgment-based Estimates of Software Development Effort

Stein Grimstad[1,2] and Magne Jørgensen[1]

[1]*Simula Research Laboratory*

[2]*University of Oslo*

Abstract

Expert judgment-based effort estimation of software development work is partly based on non-mechanical and unconscious processes. For this reason, a certain degree of intra-person inconsistency is expected, i.e., the same information presented to the same individual at different occasions sometimes lead to different effort estimates. In this paper, we report from an experiment where seven experienced software professionals estimated the same sixty software development tasks over a period of three months. Six of the sixty tasks were estimated twice. We found a high degree of inconsistency in the software professionals' effort estimates. The mean difference of the effort estimates of the same task by the same estimator was as much as 71%! The correlation between the corresponding estimates was 0,7. Highly inconsistent effort estimates will, on average, be inaccurate and difficult to learn from. It is consequently important to focus estimation process improvement on consistency issues and thereby contribute to reduced budget-overruns, improved time-to-market, and better quality software.

1. Introduction

Effort estimation is an important activity in software development and provides essential input to pricing, planning and budgeting processes [2, 4, 26]. Unfortunately, many software effort estimates are inaccurate and effort overruns seem to be the rule rather than the exception [8, 10, 21]. It is unrealistic to expect perfectly accurate estimates, even with the best estimation and development processes, since several of the factors that affect project effort can only be known after the project is completed. On the other hand, it is likely that estimation accuracy can be improved substantially by better estimation processes [1, 13]. Improved consistency in the use of effort estimation information and processes, which is the topic of this paper, is one possible approach to achieving more accurate effort estimates.

Greater consistency may, to some degree, be achieved by greater use of formal estimation models. In many other fields in which forecasts are made, such as the making of diagnoses in medicine, expert judgments are typically outperformed by even the simplest prediction models, partly due to the higher degree of consistency of the models [19]. The obvious consequence of this is that we should switch to effort estimation models instead of expert judgment in software development projects. However, the situation in software engineering seems to be different from that in many other disciplines. A recent review of sixteen studies comparing models and experts in software development effort estimation shows that the experts typically performed no worse than the models [11]. One reason for this may be that it is difficult to develop meaningful estimation models that do not require a high degree of expert judgment as input to the models in the first place; that being so, the difference between models and expert judgment-based effort estimates in software development with regard to consistency may not be large. Understanding the nature and degree of inconsistency in expert judgment may consequently benefit estimation processes based on models, as well as those based on expert judgment.

In this paper we understand “degree of inconsistency” to mean how much an individual's effort estimates of the same software task, based on the same information and made under similar conditions, but made at different times, differ. If a difference in an individual's effort estimates of the same task is caused by changed conditions, e.g., by learning or the possession of new information, the difference it is not necessarily an indication of estimation inconsistency. In psychology, this judgment inconsistency is sometimes referred to as "test-retest reliability".

Forecasting research on inconsistency, e.g., [27], suggests that inconsistency is a major source of error in forecasts based on human judgment and that it makes learning more difficult. The forecasting research has mainly been conducted in laboratory settings, which is not surprising, since people in real-life situations seldom make judgments more than once under the same conditions. In spite of the lack of real-life studies, there are good reasons to believe that there is a high degree of inconsistency in judgments made outside the laboratory. This belief is supported by, among other things, the finding that reducing inconsistency through a mechanical combination of predictions typically leads to more accurate predictions; see, for example, [9, 14, 28]. Consequently, a reduction in the degree of inconsistency in software development effort estimation may be important for improving the estimation processes.

This paper tries to contribute to this goal by providing a better understanding of the size and nature of the inconsistency in software professionals' expert judgment-based estimation. A better understanding of the degree and nature of effort estimation inconsistency may provide valuable input for the development of improved estimation guidelines, models and processes; the selection of estimation personnel; and the design of training programmes that will lead to a more consistent use of estimation information and processes. There has, as far as we know, not been any previous study that investigates empirically software professionals' degree of inconsistency in an effort estimation context. This means that we do not know the extent to which severe consequences of inaccurate effort estimates, e.g., budget-overruns, delayed time-to-market, and poor quality software, can be reduced by improving software professionals' consistency in their use of estimation information and processes.

The research questions of this paper are as follows:

RQ 1: How consistent are software professionals' expert judgment-based effort estimates?

RQ 2: Do more accurate estimators have more consistent expert judgment-based effort estimates than less accurate estimators?

The remainder of the paper is organized as follows: Section 2 briefly discusses related work on inconsistency. Section 3 describes the design of our experiment. Section 4 presents the results. Section 5 discusses the results. Section 6 summarizes.

2. Related Work

Inconsistency in expert judgment has been investigated, and demonstrated, in many research fields; see, for example, [17, 18]. One important finding is that there are considerable domain-specific differences. For example, weather forecasters are, on average, far more consistent than stockbrokers

[24]. Among professions studied with respect to consistency in judgments, none are, in our opinion, sufficiently similar to software development to enable the transfer of research results on consistency. Unfortunately, as stated earlier, we have been unable to find any empirical study of software professionals' individual level of effort estimation consistency.

It is, to some extent, understandable that there is a lack of studies on this subject. Such studies require, among other things, that software professionals estimate the effort of the same task at least twice, that they do not remember the first estimate on the second occasion, and that no significant amount of learning have taken place. These conditions can hardly be met in other situations than carefully designed laboratory conditions.

Most studies in which different software professionals estimate the same software development task report a high variation of effort estimates. In [16], for example, 14 professional software project leaders estimated the effort of the same project. The mean effort of the 14 estimates was 28 man-months. The standard deviation of the estimates was as high as 18 man-months for expert judgment-based estimates and 14 man-months for model-based effort estimates. It is, however, not reasonable to claim that this large variation in effort estimate for the same project is a proper measure of the individuals' level of inconsistency. This would require that we made several unrealistic assumptions, e.g., that the software professionals would build the same software and have the same understanding of the (typically incomplete) specification. In our opinion, analyses based on such assumptions would be highly speculative and we have, consequently, not included these studies as reference points for our own results on individuals' degree of inconsistency regarding effort estimation.

In [15] we observed that software professionals who were more optimistic on previous effort estimation tasks were the more optimistic ones on subsequent tasks in 68% of the cases. This observation suggests that there are systematic individual differences in software professionals' estimation accuracy. The opposite result, i.e., that there were no systematic difference in estimation accuracy, would suggest that the degree of inconsistency was random and that we should not expect to observe systematic individual differences in inconsistency in our experiment, i.e., the answer to RQ2 would be negative.

3. Study Design

The problems of examining inconsistency regarding effort estimation in real-life situations motivated our decision to investigate our research questions in a carefully-designed laboratory setting.

3.1 Previous Experiment

About one year before completion of the current experiment, we conducted an experiment in which 20 software professionals each estimated the effort and then completed five development tasks on an existing web-based database system written in Java [6]. The current experiment is based on the effort estimation of development tasks on the same web-based database system. The research results and study material of that previous study provide essential input to the design of the current experiment.

3.2 Selection of Subjects

We selected three software professionals with high and three software professionals with low estimation accuracy from the previous experiment as subjects for the current experiment. In addition, we selected one software professional with medium estimation accuracy, for a total of seven subjects.

All subjects are experienced software consultants with Masters degrees. They were paid for their participation. None of them had received any estimation training between participation in the previous and the current experiment. Clearly, the observation of only seven software professionals is a threat to the robustness of the results. However, for practical reasons we had to choose between a study of few subjects solving many estimation tasks or many subjects solving few tasks. We considered that our research questions were better answered with the first study design option.

Our selection of subjects ensured that they had relevant previous experience with estimation and completion of similar tasks. In addition, the nonrandom selection of the 20 subjects from the previous experiment was supposed to strengthen the analysis of whether or not the most accurate estimators were also the most consistent (RQ 2), because the difference in previous estimation accuracy among the subjects is likely to be larger, compared to that of a random selection.

3.3 Estimation Tasks

The requirement specifications are based on actual change requests from the users of the system and written in natural language. The length of the specifications varied from a few lines to a full page. An example is given below.

The current system implementation accesses the database directly. Rewrite all database code to use Hibernate for database access.

Estimate the most likely work-effort it would require for you to implement and unit test this task.

Estimate of the most likely work effort ________ (work-hours)

Two senior software developers went through the requirement specifications to ensure that there were no obvious errors in the descriptions and that it was reasonable to believe that the tasks were familiar to the subjects. The two senior software developers believed that the requirement specifications represented a typical specification met in the software industry. The only notable difference was that the specifications were more precise than the average, e.g. that there was less irrelevant information than is typical in many real-world specifications. This means that the specifications are, to some extent, estimation consistency-friendly[*]. Consequently, the degree of inconsistency may be greater in real-life situations than in our experiment.

We debriefed the subjects when they had completed all tasks. In the debriefing, the subjects stated that they had perceived the tasks as realistic and all but one subject found the estimation tasks typical for tasks they normally estimate. The outlying subject's level of inconsistency was average with respect to the studied group.

3.4 Treatment

The subjects participated in three half-day sessions with approximately one month between each session. At the start of each session, the subjects received a booklet that contained 20 estimation

[*] Forecasting research suggests that when information is presented in a way that clearly emphasizes the most relevant information, consistency improves [27].

tasks. The subjects were instructed to estimate the tasks in the same order as in the booklet. They were not allowed to go back and change previous, already completed, estimates. All seven subjects estimated the same tasks and in the same order. The tasks were estimated by expert judgment. The subjects had access to the system documentation, but not to the source code.

Six of the tasks (TT1-TT6) were used to measure the degree of inconsistency. These test tasks were estimated twice, e.g., the 10th task estimated in Session 2 was identical to the 14th task estimated in Session 1; see Table 1. Most tasks (48 out of 60) were estimated only once. This, together with the long period of time between the sessions, would imply, we assumed, that the subjects did not realize that they had estimated a test task before.

Table 1 Tasks (T1-T60) and Test Tasks (TT1-TT6)

Session 1	*Session 2*	*Session 3*
T1	T21	T41
T2 (TT2)	T22	T42
T3	**T23 (TT5)**	T43
T4 (TT4)	**T24 (TT6)**	**T44 (TT2)**
T5	T25	T45
T6	T26	**T46 (TT4)**
T7	T27	T47
T8	T28	T48
T9	T29	T49
T10	**T30 (TT1)**	T50
T11	T31	T51
T12	T32	T52
T13	T33	T53
T14 (TT1)	**T34 (TT3)**	T54
T15	T35	T55
T16 (TT3)	T36	T56
T17	T37	T57
T18	T38	T58
T19	T39	**T59 (TT5)**
T20	T40	**T60 (TT6)**

We informed the subjects that the duration of each session was stipulated to be about four work-hours, but that they could use more time, and would be paid for it, if needed. In the debriefing, six subjects reported that the time they had used on estimation was the similar to, or greater than, the time they typically used to estimate similar tasks, while one had spent less time than usual. Time pressure may increase inconsistency [23]. However, we believe that the impact of time pressure on the subject who spent less than time than usual was low. This belief is supported by the observation that this subject was, on average, the third most consistent estimator.

3.5 Measurement

To compare the estimation accuracy of the subjects in the previous experiment, we applied MRE [3] (Magnitude of Relative Error). MRE is a commonly used measure for estimation accuracy, and is calculated by the following formula:

$$MRE = \frac{\left| actual\,effort - estimated\,effort \right|}{actual\,effort} *100\%$$

As noted above, six test tasks were estimated twice. Consequently, there are 42 pairs (seven subjects * six tasks) of corresponding estimates that can be used to measure inconsistency. A pair consists of two estimates of the same development task, by the same subject, in two different sessions.

We measure the relative inconsistency (RIncons) of a pair of estimates provided by subject i on test task j by the following formula:

$$RIncons(Si,TTj) = \left(\left(\frac{\max(Est1(Si,TTj), Est2(Si,TTj))}{\min(Est1(Si,TTj), Est2(Si,TTj))} \right) - 1 \right) *100\%$$

where Si is subject i, and TTj is test task j. Est1 is the first estimate of TT j, and Est2 is the second estimate. Simplified, we measure the relative inconsistency as the ratio of the highest to the lowest effort estimate of the same task for the same subject. If, for example, S_1 estimated that he required 10 work-hours to solve TT_4 in Session 1, and 15 work-hours for the same task one month later, RIncons(S_1,TT_4) = ((max(15,10)/min(15,10) – 1) * 100% = (15/10 – 1)* 100% = 50%

The relative degree of inconsistency has the advantage that it measures degree of inconsistency independently of the size of the estimates. However, for small tasks relative degree of inconsistency can be misleading, i.e., relative degree of inconsistency can be high although the absolute difference between the estimates is of no practical importance. Therefore, we also used a measure of absolute degree of inconsistency (AIncons). We define AIncons as:

$$AIncons(Si,TTj) = \left| Est1(Si,TTj) - Est2(Si,TTj) \right|$$

4. Results

4.1 Descriptive Statistics

The estimates of the six test tasks (TT1-TT6) are presented in Table 2. We did not identify any obvious outliers, e.g., due to mistyping, in the data.

Table 2 Estimates of Most Likely Effort (work-hours)

Task	Subject 1		Subject 2		Subject 3		Subject 4		Subject 5		Subject 6		Subject 7	
	Est1	Est2	Est1	Est2	Est1	Est2	Est1	Est2	Est1	Est2	Est1	Est2	Est1	Est2
TT1	32	30	6	13	5	5	7,5	7	8	25	7	20	18	11
TT2	8	8	6	2,5	5	2	5	6	4	6	6	8	5	5,5
TT3	32	28	7	11	4	5	7	4	16	8	7	10	15	9
TT4	4	4	1	2,5	2	2	2	4	3	2	2	1	2	1,5
TT5	16	40	10	15	6	16	7	5,5	80	30	40	40	7	8
TT6	6	10	4	3	1	3	1,5	1,5	1	1	3	1	2	1

4.2 Research Question 1

We addressed Research Question 1 through examination of the difference of subjects' estimates of the same task applying the measures RIncons and AIncons; see Table 3 and Figure 1.

Table 3 RIncons (%) and AIncons (work-hours)

Subject Id	Mean		Median		Max		Min		Stdv	
	RIncons	AIncons	RIncons	AIncons	RIncons	AIncons	RIncons	AIncons	RIncons	AIncons
1	40	5,67	11	3,00	150	24,0	0	0,00	59,6	9,16
2	91	3,67	87	3,75	150	7,00	33	1,00	50,4	2,23
3	90	2,67	88	1,50	200	10,0	0	0,00	91,7	3,78
4	38	1,33	24	1,25	100	3,00	0	0,00	40,1	1,08
5	97	13,0	75	5,00	213	50,0	0	0,00	80,0	19,2
6	94	3,50	71	2,00	200	13,0	0	0,00	83,4	4,76
7	48	2,67	49	1,00	100	7,00	0	0,50	34,9	2,99
Average	71	4,64	50	2,00					66,4	8,66

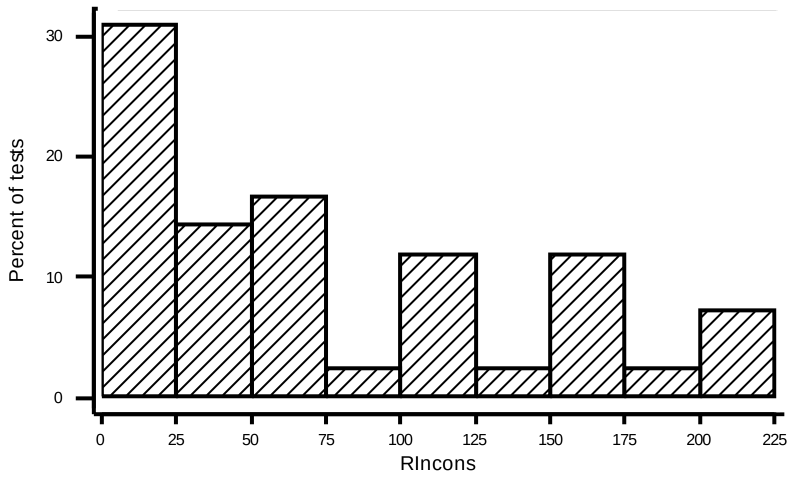


Figure 1 Histogram of relative inconsistency (RIncons)

Important observations include:

- *The degree of inconsistency is high.* Mean (median) RIncons is 71% (50%). RIncons is larger than 25% in 28 of the 42 tests. The correlation between the first and the second effort estimate of the same task is 0,7.
- *There are large individual differences in inconsistency.* The lowest mean (median) RIncons of the subjects is 40% (11%), while the highest is 97% (75%).
- *None of the subjects is consistent on all six tasks.* All subjects have RIncons of 100% or more on at least one occasion.
- *None of the tasks are consistently estimated by all subjects*. The lowest mean (median) RIncons for any test task is 54% (33%), while the highest is 86% (67%).

RIncons was, on average, higher for larger than for smaller tasks. While mean RIncons for tests tasks with both effort estimates larger than eight hours is 93%, it is 68% for test tasks with one of the estimates equal to or smaller than eight hours. This suggests that: i) the high degree of inconsistency is not explained by the fact that some of the estimated tasks are quite small, and ii) the degree of inconsistency is at least as high for large as for smaller development tasks. However, the degree of effort estimation inconsistency for large projects remains to be studied.

4.3 Research Question 2

The second research question concerns whether more accurate estimators are more consistent in their expert judgment-based effort estimates than less accurate estimators. This research question is investigated by analyzing the connection between the subjects' estimation accuracy (MRE) of the previous experiment and the degree of inconsistency (RIncons) of the current experiment. Figure 2 shows the median RIncons and the median MRE. As can be seen in Figure 2, the connection between MRE and RIncons is not strong. The correlation (r) between median MRE and median RIncons is -0,3. However, this result should be interpreted with great care, because the number of subjects is low and the impact from a few extreme observations is strong. If we, for example, remove Subject 1 (median MRE of 81% and median RIncons of 11%) from the analysis, the correlation is 0,4 in the expected direction. Subject 1 seemed to have a different estimation process than the others. The fact that one subject affects the correlation illustrates the lack of robustness of the results from this part of the study.

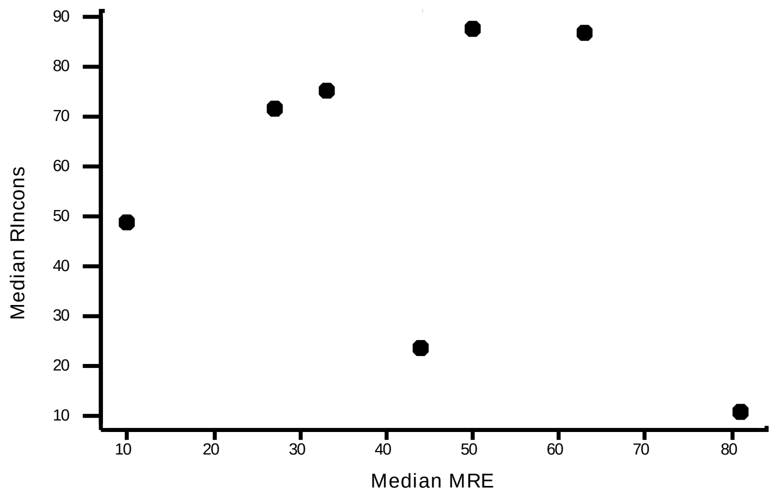


Figure 2 Inconsistency (median RIncons) vs. Accuracy (median MRE)

If it is possible to generalize from the displayed in Figure 2, it may be that we cannot select accurate effort estimators based on tests of degree of estimation consistency. However, this does not entail that estimation accuracy and inconsistency are unrelated. Clearly, individual software professionals who improve their estimation consistency are likely to improve their estimation accuracy, as well.

5. Discussion

The main result of our study is the observed high degree of inconsistency in expert judgment-based estimates. Our study was conducted in the laboratory, but we believe it is likely that real-life inconsistency is at least as large in many real-world situations. Arguments supporting our belief include the observation that inconsistency increased with larger tasks in our experiment, that the subjects perceived the experimental setting as realistic, and that the estimation tasks in the experiment in many ways were consistency-friendly. In addition, the typical higher complexity of larger projects may induce a higher degree of inconsistency.

One implication of our results is that expert judgment-based effort estimates will never be very accurate if the inconsistency problem is not properly addressed. To illustrate the impact of level of inconsistency on estimation accuracy in our study, assume that the actual effort of a task in our experiment is the mean value of the two estimates of that task, i.e., that the main source of estimation error is inconsistency and not, for example, a bias towards optimism. Then, the mean MRE is 0,2. This MRE-value indicates that the best possible level of estimation accuracy is about 20% given the observed level of inconsistency, i.e., given a median RIncons of 50%.

Clearly, this calculation has its limitations. There are, for example, factors other than the ability to provide realistic estimates that affect estimation accuracy, e.g., the ability to “develop to cost” by simplifying the process or product [5]. Such complex relationships between estimates and actual efforts make it difficult to isolate the impact of inconsistency on estimation error. Nevertheless, the high degree of inconsistency that we have measured indicates that inconsistency can, to some degree, explain the observed high estimation error in many surveys, i.e., the 30-40% development effort overrun reported in [20]. It may also explain parts of the interestimator disagreement observed in estimation studies where different software professionals estimate the same project; see the example in Section 2.

We did not find evidence that more accurate estimators are more consistent. Some possible explanations are that: 1) the power of our study was too low to examine this relationship, 2) factors other than the ability to provide realistic estimates had a strong impact on the measured estimation accuracy, e.g., variations in the ability to develop to cost, and 3) while consistency is a necessary condition for accurate estimates, it is not a sufficient condition. There are estimation methods that are perfectly consistent, but have no predictive value. If, for example, a subject responds "10 hours" every time asked for an estimate, he or she would be perfectly consistent in spite of inaccurate effort estimates.

Many explanations have been proposed for the inconsistency of human judgment in other fields; see, for example, [7, 25, 27]. Two examples of (partly overlapping) types of explanations are these: i) inconsistency is caused by the effects of presumably irrelevant variations in the decision situation, and ii) inconsistency is caused by cognitive limitations. An example of the first category is found in [25], where the weather influenced the weight that reviewers of college applications placed on academic attributes. An example of the second category is described in [22], where it is shown that a decomposing variant of a multiple-criteria decision-making technique was more consistent than a holistic variant of the same technique. However, the variety of explanations and theories also suggest that our knowledge of the causes of inconsistency is limited.

Although we know little about the causes of inconsistency, we do know something about how to reduce it. An example of a well-established method for improving consistency is to combine effort estimates from several independent estimators [9, 14, 28]. Given our observation of a high level of inconsistency, we believe that greater use of proper combination-based effort estimation would lead to substantial improvements in estimation accuracy. The empirical evidence in [12] supports this belief in the benefits derived from combining effort estimates.

Clearly, more research on inconsistency in estimation processes is needed. The degree of inconsistency we have measured needs to be validated against studies in other contexts and with larger samples and other populations, and the impact on estimation accuracy needs to be investigated further. We also need research on how inconsistency in estimation processes can be reduced, e.g., with respect to the effect of different types of estimation checklists or guidelines.

6. Summary

We reported on an experiment conducted to investigate the degree of inconsistency in expert judgment-based software development effort estimation. This is a topic that has received little attention in research on software estimation.

In the experiment, seven experienced software professionals estimated the most likely work-effort of the same 60 software development tasks. The subjects estimated six of the tasks twice, with at least one month between each estimate of the same task. We found that a subject's estimates of the same task differed substantially (mean difference 71%, median difference 50%). Highly inconsistent effort estimates will, on average, be inaccurate and difficult to learn from. Consequently, when attempting to improve processes of software effort estimation, and thereby contribute to reduced budget-overruns, improved time-to-market, and better quality software, it is important to focus on issues pertaining to consistency.

The difference in estimation accuracy of the subjects on previously completed development tasks did not predict degree of estimation inconsistency in the experiment. Possible explanations include the following: i) The power of our study was too low to examine this relationship, e.g., the correlation is as expected when removing one extreme observation from our experiment. ii) The estimation accuracy of individuals was affected by factors other than the degree of consistency, e.g., by the ability to develop to cost iii) A high degree of consistency may be a necessary, but not a sufficient condition for accurate estimates. Even if there should be a lack of positive correlation between software developers’ median estimation accuracies and median level of inconsistency, this would not gainsay our recommendation of implementing processes that will reduce estimation inconsistency. Clearly, individual software professionals who improve their estimation consistency are likely to improve their estimation accuracy.

References

[1] J. Aranda and S. Easterbrook, "Anchoring and Adjustment in Software Estimation," *proc. European software engineering conference*, pp. 346-355, 2005.

[2] L. C. Briand and I. Wieczorek, "Resource estimation in software engineering," in *Encyclopedia of software engineering*, J. J. Marcinak, Ed., 2nd ed. New York: John Wiley & Sons, 2002, pp. 1160-1196.

[3] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software engineering metrics and models*. Menlo Park, California: Benjamin Cummings, 1986.

[4] P. Coombs, *IT Project Estimation - A Practical Guide to the Costing of Software*. Cambridge: Cambridge University Press, 2003.

[5] S. Grimstad and M. Jørgensen, "A Framework for Analysis of Software Cost Estimation Error," *forthcoming in ISESE*, 2006.

[6] T. M. Gruschke and M. Jørgensen, "How much does feedback improve software cost estimation? An Empirical Study," *Paper in progress*, 2006.

[7] N. Harvey, "Why are judgments less consistent in less predictable task situations?," *Organizational Behaviour and Human Decision Processes*, vol. 63, pp. 247-263, 1995.

[8] F. J. Heemstra and R. J. Kusters, "Controlling Software Development Costs: A Field Study," *proc. International Conference on Organisation and Information Systems*, 1989.

[9] M. Höst and C. Wohlin, "An experimental study of individual subjective effort estimations and combinations of the estimates," *proc. International Conference on Software Engineering*, pp. 332-339, 1998.

[10] 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.

[11] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37-60, 2004.

[12] M. Jørgensen, " Estimation of Software Development Work Effort:Evidence on Expert Judgment and Formal Models," *forthcoming in International Journal of Forecasting*, 2006.

[13] M. Jørgensen and D. I. K. Sjøberg, "Impact of effort estimates on software project work," *Information and Software Technology*, vol. 43, no. 15, pp. 939-948, 2001.

[14] M. Jørgensen and K. Moløkken, "Combination of software development effort prediction intervals: Why, when and how?," *proc. Conference on Software Engineering and Knowledge Engineering*, 2002.

[15] M. Jørgensen, B. Faugli, and T. M. Gruschke, "Characteristics of Software Engineers with Optimistic Predictions," *forthcoming in Journal of Systems and Software*, 2006.

[16] R. J. Kusters, M. J. I. M. Genuchten, and F. J. Heemstra, "Are software cost-estimation models accurate?," *Information and Software Technology*, vol. 32, no. 3, pp. 187-190, 1990.

[17] K. Levi, "Expert systems should be more accurate than human experts: Evaluation procedures from human judgment and decision making," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 3, pp. 647-657, 1989.

[18] C. M. Lusk and K. R. Hammond, "Judgment in a dynamic task: Microburst forecasting," *Journal of Behavioral Decision Making*, vol. 4, pp. 55-73, 1991.

[19] P. E. Meehl, "When shall we use our heads instead of the formula?," *Journal of Counseling Psychology*, vol. 4, no. 4, pp. 268-273, 1957.

[20] K. Moløkken and M. Jørgensen, "A review of software surveys on software effort estimation," *proc. International Symposium on Empirical Software Engineering*, pp. 223-230, 2003.

[21] K. Moløkken-Østvold, M. Jørgensen, S. Tanilkan, H. Gallis, A. Lien, and S. Hove, "A Survey on Software Estimation in the Norwegian Industry," *proc. Metrics '04*, pp. 208-219, 2004.

[22] O. F. Morera and D. V. Budescu, "Random Error Reduction in Analytic Hierarchies: A Comparison of Holistic and Decompositional Decision Strategies," *Journal of Behavioral Decision Making*, vol. 14, pp. 223-242, 2001.

[23] H. G. Rothstein, "The effects of time pressure on judgment in multiple cue probability learning," *Organizational Behaviour and Human Decision Processes*, vol. 37, pp. 83-92, 1986.

[24] J. Shanteau, D. J. Weiss, R. Thomas, and J. Pounds, "Performance-based assessment of expertise: How can you tell if someone is an expert?," *European Journal of Operations Research*, vol. 136, pp. 253-263, 2002.

[25] U. Simonsohn, "Clouds Make Nerds Look Good: The Impact of Environmental Cues on Attribute Weighting," *forthcoming in Journal of Behavioral Decision Making*, 2006.

[26] Sommerville, *Software Engineering*, 7. ed: Addison-Wesley, 2004.

[27] T. R. Stewart, "Improving Reliability of Judgmental Forecasts," in *Principles of Forecasting*, J. S. Armstrong, Ed. Boston: Kluwer Academic Publishers, 2001, pp. 81-106.

[28] L. M. Taff, J. W. Borchering, and J. W. R. Hudgins, "Estimeetings: development estimates and a front-end process for a large project," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 839-849, 1991.

Paper III:

The Clients' Impact on Effort Estimation Accuracy in Software Development Projects

Stein Grimstad[1,2], Magne Jørgensen[1] and Kjetil Moløkken-Østvold[1,2]

[1]*Simula Research Laboratory*

[2]*University of Oslo*

Abstract

This paper focuses on the clients' impact on estimation accuracy in software development projects. Client related factors contributing to effort overruns as well as factors preventing overruns are investigated. Based on a literature review and a survey of 300 software professionals we find that: 1) Software professionals perceive that clients impact estimation accuracy. Changed and new requirements are perceived as the clients' most frequent contribution to overruns, while overruns are prevented by the availability of competent clients and capable decision makers. 2) Survey results should not be used in estimation accuracy improvement initiatives without further analysis. Surveys typically identify directly observable and project specific causes for overruns, while substantial improvement is only possible when the underlying causes are understood.

1. Introduction

Overruns in software development projects have been a major concern for several decades. The topic of effort estimation has been given much attention in software engineering research, for instance by the development of numerous algorithmic estimation models. Still there is no evidence of improvement in effort estimation accuracy over the last 20 years. A recent review of estimation surveys [1] shows that most surveys of effort estimation performance in software development projects report average overruns of 30-40%. Even if it is debatable whether the magnitude of these overruns constitutes a 'software crisis' [2], it is likely that organizations who improve estimation accuracy will gain a competitive advantage, e.g., as a consequence of increased predictability and more optimal resource allocation. This applies to clients as well as vendors, since disadvantages caused by overruns hit both clients and vendors, irrespective of contract type, as discussed in [3].

It is often claimed that clients, as well as vendors, influence effort estimation accuracy in software development projects. Previous research supports this view. For instance, a survey of estimation overruns in software development projects showed that governmental projects on average had higher overruns of effort estimates than private projects [4]. The governmental and private projects did not differ on any of the project characteristics measured, such as development methodology, vendor capability, project size, or project duration. The only difference was the category of client.

By focusing solely on clients' impact on estimation accuracy and by investigating both negative and positive impact clients have on estimation accuracy we hope to gain a deeper understanding of this relationship. For clients such understanding is important in order to improve their acquisition processes. Besides, an understanding of their own capability as clients of software development projects might be valuable input to the selection of vendors. The understanding of clients' impact on estimation accuracy is also valuable to vendors. Realizing that the client is the cause of a troubled project enables appropriate actions to be taken. These might include changes to the development methodology, the project staffing, re-negotiations, juridical actions, etc. Note that the term 'client' is used in a broad sense in this paper; we have not made any attempt to distinguish between 'clients', 'users' and 'customers'.

The goal of this study is to examine 1) Do clients influence estimation accuracy? 2) How do clients impact estimation accuracy? 3) What can clients do to improve estimation accuracy? These questions are examined by reviewing software engineering literature and surveying software professionals. The paper is organized as follows: Section 2 presents a review of existing research on how customers impact estimation accuracy in software development projects. That review is the background for the study design described in Section 3. The results of the study are presented in Section 4, while Section 5 discusses the results of the study and the findings in the review with respect to the objectives of the study. Section 6 concludes the paper.

2. Review of Related Work

The aim of the review is to identify research relevant to the research questions presented in Section 1 in an unbiased and auditable manner.

2.1 Review Design

The design of the review is based on the guidelines for structural reviews in software engineering proposed by Kitchenham [5]. The review is limited to what we consider the most important journals for software engineering; Information and Software Technology, Journal of Systems and Software, IEEE Transactions of Software Engineering and ACM Transactions on Software Engineering and Methodology. Papers were selected for inclusion by manually going through the online indexes of these journals, reading the abstracts that appeared relevant and finally the full versions of the remaining papers. The review was done by one of the researchers. For each included paper, the following information was extracted: the bibliographic information, study type, goals of the study, sample size and the client's impact on estimation accuracy. The criteria used to decide whether to include or exclude papers were:

- the papers have to report causes or risk factors for estimation accuracy/performance/overruns or project success/failure (when success/failure was related to schedule/budget or estimates)
- the papers have to report empirical evidence from software development projects
- the papers have to identify general reasons (as opposed to investigate a specific issue)
- if two studies are based on the same data set, only one were included
- the papers have to be available online (in most cases, papers from the last 10-15 years were available)

There are a number of limitations to this review. Among the most important is that the survey is based on a limited selection of journals. We are aware of other journal papers, books, conference papers and industry reports that have made important contributions to the topic such as Brooks' classic The Mythical Man-Month [6]. We have chosen this approach in an attempt to get an unbiased sample of material.

There is also a risk that papers from the selected journals have been wrongly excluded, i.e. that we were not able to recognize the relevance of the paper by the name and/or the abstract. Some verification was done by checking against a previous review [7].

Another risk is the subjective identification of client influenced factors. This task is hard as the categories/terms/reasons are most often only explained in broad terms. We have not found any good way to address this risk other than to rely on our experience as researchers and software professionals. There is also the problem that papers built on considerable experience are excluded from this review as they do not directly report empirical evidence. However, inclusion of such articles would lower the quality of the review, as it would be even more subjectively decided which articles to include and which to exclude.

2.2 Review Results

We identified eight studies that matched our criteria.

2.2.1 Verner et al. [8]. Verner et al. [8] conducted structured interviews with 20 senior software development professionals from a number of different organizations in the USA. The goal of the study was to compare the software project management advices given in Brooks' famous book the Mythical Man-Month with practices employed 25 years later. For each topic discussed, the factors leading to project success were identified, and then how the same factors could contribute to failure. The client related success factors found, and their frequency, are: High level management support (about 50%), customer and user involvement (15%), good requirements (nearly 50%), flexibility (frequency not available) and communication (67%). The corresponding failure factors are: Lack of higher level management support (almost all), lack of involvement/confidence and too many customers involved (nearly 50%), vague/poor requirements and no clear vision (40%), poor estimates made by management and dictated dates (50%) and feature and scope creep (reported as "many" in the paper).

2.2.2 Procaccino et al. [9]. Procaccino et.al. [9] had 21 IT professionals reflecting 42 software development projects complete two questionnaires (failed/successful projects). The goal of the study was to investigate some of the most influential success factors early in the development process. The respondents were project leaders, technical support personal and developers. All were from the same organization. The respondents were asked what they perceived as success factors, and what they thought management considered as success factors (there were differences). The significant success factors, when applying Chi-square tests, were: Presence of a committed IT sponsor, customer/user's involvement/commitment/confidence in the project, customer/users involvement in schedule estimation, customer/users had realistic expectations, establishment of complete and accurate requirements and customers/users allocated adequate time for requirement gathering.

2.2.3 Lederer and Prasad [10]. Lederer and Prasad [10] used a questionnaire to have 112 systems managers and other information systems professionals rate 24 predefined reasons for overruns. The customer factors perceived as important out of the ten most important are: Change requests by users, users' understanding of requirements, user-analyst communication and understanding, poor or imprecise problem definition and coordination of company functions during development. The causes where correlated with the organization's percentage of inaccurate estimates by use of the Pearson r coefficient correlation. The customer factors of the top ten most statistic significant reasons are: Reviewers don’t consider whether estimates are met, lack of careful examination of the estimate by management and poor or imprecise problem definition.   .

2.2.4 van Genuchten [11]. van Genuchten [11] performed weekly interviews with project leaders and collected data on activity level for six software development projects in the same software development department in order to gain an insight in the reasons for delays of software development projects. The most frequent reasons for delays were "more time spent on other work than planned" (mainly as a result of maintenance tasks) and "complexity of application underestimated". None of these are customer related. Measurements in other departments revealed that distribution of causes varies strongly for each department.

2.2.5 Jiang and Klein [12]. Jiang and Klein [12] had 86 members of Project Management Institute complete a questionnaire to test the linkage between previously identified software development risks and various dimensions of system success. Only two of the risks had a significant impact on "meet budgets" and "meet schedules": "Lack of teams general expertise" (including the ability to work with uncertain objectives, ability to work with top management and ability to understand human implications of a new system) and "Lack of role clarity" (including the role of each person involved in the project is not clearly defined and communications between those involved in the project are unpleasant).

2.2.6 Ropponen and Lyytinen [13]. Ropponen and Lyytinen [13] surveyed 83 project managers from the Finnish Information Processing Association (at most two respondents from each company) using a questionnaire. The goal of the study was to investigate the impact of risk management practices on software development. Schedule and timing risks were influenced by the following client related factors: The project size (larger projects performed worse than smaller), the client's industry (retail business, accommodation, nutrition performed better than other industries) and the application type (interactive applications performed worse than other).

2.2.7 Jørgensen and Moløkken-Østvold [7]. Jørgensen and Moløkken-Østvold collected data from estimation experience reports of 68 projects and interviewed eight employees (in different roles) in a Norwegian software company. The goal of the study was to understand how roles, information collection approaches and analysis techniques supplement each other when examining reasons for errors in software effort estimates. Reasons for estimation error mentioned in the interviews were: Lack of realism in HCI-requirements, lack of requirement change control processes, unrealistic expectations by clients and lack of good requirement specifications leading to unplanned re-work. Reasons for estimation inaccuracy found in the experience reports where: Change requests from clients or "functionality creep", resource allocation problems, poor requirement specifications or problems with communication with the client or that high priority where on quality and not on cost accuracy. Reasons for estimation accuracy found in the experience reports where: Simple projects and a high degree of flexibility in how to implement the requirement specification. The study also contains a statistical analysis (stepwise regression) of some project characteristics and how they relate to estimation accuracy. The only significant client factor was the client's priority of time-to-delivery.

2.2.8 Subramanian and Breslawski [14]. Subramanian and Breslawski [14] had 40 members of the ACM Special Interest Group on Software Engineering responded to a mail questionnaire. The study seeks, among other things, to explain the percentage of relative error in software effort estimation. The customer influenced reasons for failure were: "requirements change/addition/definition", "design changes, scope and complexity", "upper management influence, bidding and time constraints"

2.2.9 Summary. None of the studies identified in this review explicitly investigated client influence on estimation accuracy in software development projects, but seven out of eight studies report that client factors are perceived as important for estimation accuracy. This strongly suggests that clients influence estimation accuracy. Factors related to management, communication and involvement in the project along with factors related to requirements and realistic expectations are the most frequent reasons perceived as impacting estimation accuracy that can be attributed to clients. Other client related factors, such as project size, the industry of the client, application type and flexibility are also reported, but less often. Table 1 summarises our interpretation of the most frequent client factors impacting estimation accuracy reported in the reviewed studies.

The studies further report that vendors are more likely to attribute failure than success to clients [8] and that who you ask (developers, project managers, management, etc) influences the perception of the clients' impact on estimation accuracy [8, 9]. Also, statistically analysis of factors give different results than surveys and interviews of what reasons are perceived as most important [7, 10]. One of the papers finds reasons for estimation accuracy to be largely project specific. When the results of a case study conducted in one department [11] was compared to results from another department within the same organization, there were significant differences.

However, it is not clear what clients should do to improve estimation accuracy in software development projects. Our study, described in the remaining sections of this paper, explicitly investigates the clients' contribution to estimation accuracy. The aim is to get more data regarding these phenomena so that we can do a more thorough analysis and better understand them.

Table 1 Client factors frequently reported to affect estimation accuracy

Factor	Study
Management	[7-10, 12, 14]
Communication	[7, 8, 10, 12]
Involvement in project	[8-10]
Requirements	[7-10, 14]
Realistic expectations	[7-9, 14]

3. Survey Design

To investigate our research questions, we conducted a survey at a technical conference. Sections 3.1-3.3 describe the design of the schema and the sample, while sections 3.4 and 3.5 comment the data collection and the analysis.

3.1 General

The findings in some of the studies in Section 2, along with our previous research and industry experience, determined the content in the survey instrument we used to investigate the research questions described in Section 1. The survey collected three types of data; 1) context information about the respondents (such as estimation experience and project roles) 2) the respondents' perception of clients' impact on estimation accuracy (positive and negative) and 3) the estimation performance in the respondents' last completed project along with their rating of a predefined set of client factors.

3.2 Schema Design

Four software professionals validated the schema in a pilot study. This led to clarification of some questions. The final schema consisted of 20 multiple choice questions and two open-ended questions. The schema was written in Norwegian and the data are later translated by us. The motivation for using Norwegian, and thus introducing the risk of translation errors, is that we believe this would lower the burden of participation and therefore increase the number of respondents. This approach would only exclude the non-Scandinavian speakers since Norwegian is quite similar to Swedish and Danish. The schema took approximately 10 minutes to complete. To motivate participation, the respondents could win prices in a lottery. In order to participate in the lottery, the respondents had to write their email address, but participation in the lottery and hence writing the email address was voluntarily. However, all respondents chose to write their e-mail address. A threat to this type of data collection is the risk of misunderstandings. The main strength of using a survey instrument, compared to interviews, is that the number of data points increase.

The respondents decided themselves how to interpret "estimate", "overruns of estimates" and "no, or small, overrun of estimates". Previous studies has shown that the estimation terminology in use is ambiguous, i.e. the term estimate is frequently used for most-likely estimates, estimates of budgets and price to customer [15]. In this study we have chosen to use broad terms such as "overruns" and "estimate". We did this because we did not want to confuse the participant with unfamiliar terms, and since we believe that the reasons for overruns are the same for the different meanings of the term "estimate". Similarly, we did not specify whether the terms "estimate" and "overruns" referred to cost, effort or schedule, as we believe that fairly similar factors cause overruns for these different types of estimates. To measure the magnitude of overrun in their last completed projects we used broad categories that were further combined in the analysis. We asked for information about the most recent project, instead of "average values" or a "information about a typical project", to reduce the influence of poor memory and selection bias.

3.3 Sample

The survey was conducted at the JavaZone 2004 conference in Oslo, Norway. The conference was arranged by the Norwegian java user group (javaBin). JavaZone targets Scandinavian professionals with an interest in Java technology, and is one of the leading technical conferences in Scandinavia. 800 persons registered for the conference including speakers, journalists, organizing committee and expo personnel. 307 of these participated in the survey. Table 2 presents some demographic data collected. The first column states the questions the respondents answered, the second shows the categories the respondents could choose from, the third column presents the frequency of each category and the percentage of the total (excluding non-respondents) while the last column contains comments. The data suggest that the respondents have a technical focus (91,5% has programmed in at least one project), they are fairly experienced estimators and that they are distributed across industries. It is also reasonable to believe that the attendants are above average interested in software development since they attend the conference, and that the average level of competence is high since most of the talks at the conference were at an advanced level.

Table 2 Demographic data

Question	Category	Result	Comment
How many projects have you participated in estimating?	None	27 (9,0%)	N = 300.
	1 – 4	113 (37,7%)	
	5 – 20	118 (39,3%)	
	20 +	42 (14%)	
Has the projects you have participated in estimating been for an internal or an external client?	Not estimated	25 (8,3%)	N = 300. Nine respondents gave multiple answers. These are interpreted as "equally many". One respondent did not answer the question, but had answered "no estimation experience" on a previous question. This is interpreted as "has not estimated".
	Internal	74 (24,7%)	
	External	126 (42%)	
	Both internal and external	75 (25%)	
	Other	0 (0%)	
What sector have you primarily been estimating for?	Not estimated	26 (8,7%)	N = 299. Each respondent were allowed to select several industries. The "other" category was mainly retail. 222 respondents (74,2%) had estimated for more than one sector.
	Governmental	93 (31,1%)	
	Telecom	96 (32,1%)	
	Financial	97 (32,4%)	
	Industry	46 (15,4%)	
	Other	43 (14,4%)	
What has been your role(s) in the above mentioned projects?	Developer	271 (91,8%)	N = 295. Each respondent was allowed to select several roles. All the non-respondents replied "not estimated" on the questions above.
	Architect	141 (48,0%)	
	Project manager	62 (21,0%)	
	Other	6 (2%)	
In your latest completed project, how large was the overrun of estimates?	None	57 (19,2%)	N = 297. 80% of the projects had overrun of estimates which is similar to other estimation surveys [1].
	0 – 20%	133 (44,8%)	
	21 – 50%	55 (18,5%)	
	Above 50%	18 (6,1%)	
	Do not know	34 (11,5%)	
In your latest completed project, what was the budget (approximately)?	Do not know	40 (13,4%)	N = 298. The figures are in Norwegian Kroner (NOK). 1 US Dollar is approximately 6,3 NOK.
	< 100.000	28 (9,4%)	
	100.000 – 1 million	86 (28,9%)	
	1 – 10 millions	102 (34,2%)	
	Above 10 millions	42 (14,1%)	

3.4 Data Collection

The survey was handed out from the organizers' stand in the conference expo area. It was labeled as a joint effort between the University of Oslo and the Norwegian java user group. A research assistant distributed and collected the schemas, and the survey was open for participation from the start of the conference till the social program started, a total of 12 hours. About 400 schemas where handed out and 307 of these were returned.

We were unable to collect reasons for non-participation in a systematic way, as the organizers declined our request to include questions regarding survey participation in the official conference evaluation form. To collect data on non-participation, we asked conference attendees during the social program about their participation. Three reasons for non-participation dominated: 1) The conference program was too packed so they did not prioritize participation in the survey, 2) They did not visit the expo area and therefore were not aware of the survey, and 3) They did not feel qualified to answer (marketing staff, journalists, etc). This means that there seems to be no harmful bias regarding participation in the survey as would have been the case if for instance one company refused to let its employees participate or if people where ashamed to participate because they had a bad track record for estimation performance.

3.5 Analysis

The data were analyzed and structured using Excel and Minitab. Seven out of the schemas were excluded from the analysis because we regarded them to be of insufficient quality (primarily because they were mostly blank). Preliminary results from the study were verified at two of the monthly member meetings in the java user group. The attendants at the meetings agreed with the interpretations.

The data regarding participants' perception of the customers' influence on estimation accuracy were categorized. The categories were derived by analyzing the answers and joining answers into groups until we had a sufficiently small number of categories. The categorization was done independently by one of the authors and an experienced software professional. Disagreements were discussed. In the few cases were an agreement was not reached, both categories were included. For the completed projects, overrun magnitude was collected by asking the respondents to choose the category that best fitted their project. They could choose from: "no overrun", "0-20%", "21-50%", "more than 50%" and "do not know". In the analysis, "no overrun" and "0-20%" is merged together as "< 20% overrun", while "21-50%" and "more than 50%" is merged together as "> 20% overrun". 73 of the responds were in the category "> 20% overrun" and 189 in the category "< 20% overrun".

To preserve anonymity, we did not ask for any project identification. This is unlikely to be a problem for the part of the survey that is concerned with the respondents' perception of factors that influence estimation accuracy, as this address the respondents opinions based on their total experience. However, for the rating of factors in the last completed project, we do not know how many projects (experimental units) the respondents represent. The data suggests that the distribution is fairly good (based on combining estimation accuracy, size of project and type of customer), but it is likely that at least some projects are overrepresented.

4. Results

4.1 Software Professionals' Perception of Client Factors Impacting Estimation Accuracy

This section presents the respondents perception of client factors that frequently caused overruns and client factors that prevented overruns. The questions asked were as follows (translated):

Q1:"In projects where estimates have been overrun, which client related factors have contributed to the overrun?"

Q2:"In the projects where estimates were not overrun, or there were only minor overruns, which client related factors contributed to prevention of overrun?"

The responses were grouped in categories, see table 3, and are presented in table 4.

Table 3 Categories of perceived reasons

Code	Category
STAB	Requirement changes and new requirements
REQU	Well defined requirements
COMM	Client – vendor communication
FLEX	Flexibility in the project (give and take)
TECH	Integration with technical environment, infrastructure and development environment
SUBC	Integration and co-operation with 3rd party vendors
SIZE	Project size
REAL	Realistic expectations (requirements, time, budget, etc)
SKILL	Availability of competent customers and capable decision makers
PROJ	Project administration and steering
OTHE	Other reasons that we were not able to classify

Table 4 Customer factors impacting estimation accuracy

Factor type	Perceived as causing overruns by	Perceived as preventing overruns by
STAB	118	23
REQU	97	50
COMM	13	34
FLEX	9	28
TECH	17	3
SUBC	4	2
SIZE	0	4
REAL	15	13
SKILL	70	76
PROJ	38	50
OTHE	14	18

Out of the 300 replies included in the analysis, 48 respondents did not mention any client factors contributing to overruns (Q1), while 89 respondents did not mention any client factors preventing overruns (Q2). In the cases where the same respondents had made several answers that fell into the same category, each answer was counted.

The three client related reasons most frequently perceived as contributing to overruns are 1) frequent requirement changes and new requirements, 2) lack of well defined requirements and 3) lack of competent customers able to make decisions. The most important reasons perceived as preventing overruns are 1) competent clients able to make decisions, 2) well defined requirements and 3) adequate project administration.

Table 5 Factors tested for connection with overruns

Code	Factor	Statement type	Translated statement/question
METH	Project Methodology	Neutral	The project used an incremental/iterative development method.
REAL	Realism in plans and budgets	Positive	The project had realistic plans and budgets
GOAL	Clear project goals	Positive	The goals of the project were clearly defined and communicated.
PRIO	Client's priority of the project	Positive	The project had high priority in the client organization.
RESO	Client's resource allocation	Positive	The client had allocated sufficient resources for an efficient project execution (test environment, end-users, etc).
SKILL	Client skills	Positive	The clients had the right skills for an efficient project execution.
COMM	Client and vendor communication	Positive	The communication between client and vendor were adequate.
STAB	Scope creep	Negate	The requirement specification were frequently expanded.
FLEX	Project flexibility	Positive	The project had the flexibility to reduce scope (functionality/quality) in order to meet plan and budget.
REWO	Client's change of opinion	Negative	Clarifications made by the client were later changed so that work had to be re-done or thrown away.
UNFO	Unforeseen tasks	Negative	Unforeseen tasks occurred frequently
PARA	Projects run in parallel	Negative	The project were delayed by projects running in parallel.
LUCK	Luck/bad luck	Negative	Luck or bad luck had a significant impact on the outcome of the project.

4.2 Correlation Between Client Factors and Effort Overruns in the Respondents' Last Completed Project

The respondents were asked to rate their last completed project (from 1 = "totally agrees" to 5 = "totally disagrees") according to a set of predefined statements. Table 5 describes the statements the respondents rated (column four) and the associated factors the statements were intended to test (column two). The statements have a mixed framing strategy. Column three describes whether the statement is positively or negatively framed. For the positively framed statements, low scores (that is, agreement) are believed to be better, while high scores (disagreement) are believed to be better for negative framed statements.

Table 6 presents the average ratings for each factor. The first column in Table 6 lists the different factors' by code. The two next columns show the estimation performance (divided into above or below 20% overrun), while the last column shows the difference between the groups. The average is calculated after removing the responses in "Don’t know" and "No response". 21 was the highest count removed for any factor. We report mean values instead of median values as we believe this gives a better picture of the dataset (no extreme values and fairly uniformly distributed values).

The projects with large overruns differ most from the projects less overruns for 1) realism in plans and budgets, 2) project flexibility and 3) client's change of opinion. The clients' priority of the project is the only factor where there is almost no difference.

Table 6 Factors connected with estimation performance

Project outcome	Less than 20% overrun	More than	Difference
METH	3,58	3,90	-0,32
REAL	2,74	3,99	-1,25
GOAL	3,46	3,89	-0,43
PRIO	4,24	4,25	-0,01
RESO	3,92	3,42	0,50
SKILL	3,26	3,55	-029
COMM	3,34	3,85	-0,51
STAB	4,25	3,77	0,48
FLEX	3,07	3,68	-0,61
REWO	3,15	2,63	0,52
UNFO	3,82	3,37	0,45
PARA	3,15	2,89	0,26
LUCK	2,44	2,03	0,41

5. Discussion of Results

5.1 Do Clients Influence Estimation Accuracy?

252 out of 300 software professionals (84%) mention at least one client factor that they considered to be a major contribution to estimation overrun. Similar, 211 out of 300 (70%) listed one or more client factors contributing to prevention of overruns. Also, the respondents' projects with the largest

overruns have worse average rating for all the rated factors (on their last completed projects), with the exception of "Client's priority of the project" which is virtually identical for both categories of projects. The importance of the client impact on estimation accuracy is further supported by related work (see Section 2.2.9).

It seems obvious that clients can impact vendors' estimation accuracy, e.g. by frequently changing requirements. The results reported in this study, and in related work, suggest that clients are perceived by software professionals as major contributors to estimation overruns. But these studies reports the perceptions of the respondents, and not objective facts, so the extent of clients' impact on vendors' estimation accuracy is unclear. There might be a number of reasons why respondents blame clients for overruns undeservedly (some of these reasons are discussed in the following section).

5.2 How do Clients Impact Estimation Accuracy?

The factors perceived as influencing estimation accuracy in this survey correspond well to factors reported in earlier surveys in the sense that at least one other study mention a category similar to each category identified in this survey as a major contributor to effort overruns (see Section 2.2.9). Still, even if there is a great deal of overlap between the surveys, none of them have provided the exact same list of factors causing overruns. These differences can probably be explained by the different focuses of the studies, terminology ambiguousness, the roles of the respondents, the method of analysis and the size differences in the samples.

Almost all of the client factors perceived by the software professionals as influencing estimation accuracy are project specific and direct causes, and little focus is on more general and underlying causes (see Table 4). For instance, "requirement changes and new requirements" is perceived as important, but less focus is on why scope is increased (for instance, "business changes") or why this caused an overrun (for instance, "the project buffer was too small"). Studies by Procaccino et al. [9] and Jørgensen and Moløkken-Østvold [7] suggest that the role of the respondents influence type of answers. Both these studies have observed that project participants are likely to provide different reasons for estimation inaccuracy than managers do. Even if the underlying reasons might be most valuable, it is unrealistic to expect that the participants in a brief survey should report such reasons when asked open-ended questions.

The perception of the factors that cause overruns corresponds to the perception of factors that prevent overruns in the sense that the same categories are suitable for grouping the respondents' answers (see Table 4). However, there are some substantial differences in the weights the factors are given. While "requirement changes and new requirements" is perceived as the most frequent contributor to overruns, only 23 mentions this as a factor that prevents overruns. Similarly, 97 mentions (lack of) "well defined requirements" as a factor causing failure, while only 50 mention it as a reason for success. The most important success factor ("availability of competent customers and capable decision makers ") mentioned rank third of the factors causing failure. A possible explanation for these differences is the human tendency towards biased self-assessment, i.e., "we made the project a success" while "they made the project a failure". In Verner et al. [8] more than half of the projects blamed clients for overruns, while only a small fragment credited the client for success. Several of the factors attributed to the client, such as client-vendor communication and integration and co-operation with 3rd party vendors, might in reality be factors mostly influenced by the vendor. An inherent limitation of this kind of surveys is that we only explore what the software professionals think and recall. It is, for instance, easier to note the presence of something than the absence. Issues, such as client-vendor communication, might happen at a level that is not visible to the respondents in our survey.

Even if most of the perceived reasons for overruns are not directly tested in the connection between factors and estimation performance (see Table 6), the perceived reasons for overruns have related factors that rate worse for the projects with bad estimation performance. However, the frequency of the perceived factors does not correspond with the impact of the factors in the rating. The most frequent factor perceived as causing overruns is "requirement changes and new requirements". Even though the corresponding factor rates worse for projects with overruns than for projects without, the rating is no worse than for other factors corresponding to reasons that are much less frequently mentioned as contributing to success or overruns. This effect is also found in Lederer and Prasad [10] and in Jørgensen and Moløkken-Østvold [7].

It seems as if clients are perceived as impacting estimation accuracy in many ways. However, the results obtained in surveys will vary according to several factors, e.g., the questions asked (overrun prevention vs. contribution), who you ask (developers, managers, etc) and the method of analysis applied (statistical analysis vs. respondents' perceptions). Still, there seems to be a limited number of symptoms of troubled projects that point to client specific causes. These symptoms are: a) increased scope b) increased complexity c) waiting, d) lack of control e) re-work f) lack of incentives.

5.3 What Can Clients do to Improve Vendors' Estimation Accuracy?

Before any action is taken to change client behavior to improve estimation performance, it should be investigated whether there actually is an estimation accuracy problem that needs to be solved. For instance, there is no estimation problem if inaccuracy was caused by a deliberate choice (i.e., extended functionality to increase return of investment) or if it was caused by planned events that are unlikely to re-occur (i.e. organizational changes). However, in most cases, we believe, estimation inaccuracy is undesirable. If analysis shows that the troubled projects have been suffering from any of the symptoms described in Section 5.2, clients might be contributing to the inaccuracy. But the solution is rarely as simple as that the client should just stop doing whatever actions leading to the causes identified. In order to obtain sustainable improvements, we need to identify the *underlying causes* [16]. Example 1 illustrates how one of the most frequently mentioned client reason for overrun can have several different underlying causes.

EXAMPLE 1: Let us assume that a company suffers from effort overruns in their software development projects, and that an analysis of the experience reports has identified that "frequent change requests during development" seems to be the most prominent factor causing problems. So, does this mean that the client should stop making change requests? In order to answer that question we need to understand why these change requests occur. Some possible answers are:

1) The client tries to get more functionality than originally agreed.
2) The original requirements were wrong and/or incomplete, so that the originally specified solution will have no or little value.
3) Unexpected business changes demand changes to the solution.

All the above mentioned reasons are possible causes for "frequent changes to the requirement specification". But they point to different underlying problems, and require different actions to eliminate. A possible solution for the first issue can be introduction of contracts that enforce better change management. The second and third issues require further analysis. Was it because insufficient time was spent on making the specification? Did the wrong people work on it? Was it impossible to know the requirements up front? In order to efficiently remove the problem, we need to understand the answer to these questions.

Another way to attack the problem of changing requirements would be to address the conditions that make the changes harmful. Table 6 shows that even if the projects that had large overruns generally rated worse on requirement stability, more than half of the successful projects (projects with overruns less than 20%) reported that they had unstable requirement A previous study by one of the authors [17] found that while the projects experience reports implied that one of the most important reasons for overruns were incomplete requirement specifications, comparisons of the requirement specification information and the estimation precision indicated the opposite! The reason for this counter intuitive result may have been that incomplete requirement specification typically meant more flexible deliveries. Therefore, understanding which circumstances make "changes to requirement specification" cause estimation overruns, is essential in order to improve the situation.

Similar examples could easily be created for most of the factors perceived as causing overruns reported in our survey. Such examples illustrate the danger of relying on individuals' perceptions in estimation accuracy improvement initiatives. A consequence is that surveys on reasons for estimation inaccuracy, project success, overruns, etc. should solely be used as a starting point for further analysis. As discussed in Section 5.2, an estimation survey does not report an objective truth and the identified causes depend on several factors. Therefore, a first step in an estimation improvement initiative might be to investigate your own projects to determine which factors are relevant for your organization. This should be followed by an in-depth analysis, for instance by applying structured interviews and investigating project artifacts, to determine the underlying causes of the problem. It is important to note that such an analysis should investigate both the actions that cause problems, and the conditions that enable the problem to arise. Once such understanding is obtained, sustainable improvement can be made.

6. Conclusion

This study suggests that clients are perceived by software professionals as impacting estimation accuracy. In a survey of 300 software professionals, the respondents answered that the most common contributions to estimation inaccuracy by clients are frequently changing, and new, requirement along with the lack of well-defined requirements and the absence of competent customers and capable decision makers. Overruns are prevented when competent customers and capable decision makers are present, the project administration and steering is adequate and requirements are well-defined. However, the results of such surveys have limited value for an organization's estimation accuracy initiatives since the factors identified in surveys typically are project specific and "direct" causes, and not the underlying causes necessary to eliminate the problem. Also, the survey results strongly depend on a number of factors such as the data collection approach, method of analysis and the framing of the questions. Despite these limitations, surveys on factors impacting estimation accuracy might be valuable as a starting point for more thorough analysis of factors impacting estimation accuracy.

References

1. Moløkken, K. and M. Jørgensen. *A review of software surveys on software effort estimation*. in *International Symposium on Empirical Software Engineering*. 2003. Rome, Italy: Simula Res. Lab. Lysaker Norway.
2. Glass, R., *Failure Is Looking More like Success These Days.* IEEE Software, 2002. **January/February**: p. 103-104.
3. Jørgensen, M. and S. Grimstad. *Over-optimism in Software Development Projects: "The winner's curse".* in *CONIELECOMP*. 2005. Puebla, Mexico.
4. Moløkken-Østvold, K., et al. *Management of Public Software Projects: Avoiding Overruns.* in *Hawaiian International Conference on Business*. 2005. Hawaii, USA: www.hicbusiness.org.
5. Kitchenham, B., *Procedures for Performing Systematic Reviews*, in *Keele University Technical Report TR/SE-0401*. 2004, Keele University: Keele.
6. Brooks, F., *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. 1995: Addison-Wesley Professional.
7. Jørgensen, M. and K. Moløkken-Østvold, *Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method.* IEEE Transactions on Software Engineering, 2004. **30**(12): p. 993-1007.
8. Verner, J.M., S.P. Overmyer, and K.W. McCain, *In the 25 years since The Mythical Man-Month what have we learned about project management?* Information and Software Technology, 1999. **41**: p. 1021-1026.
9. Procaccino, J.D., et al., *Case Study: factors for early prediction of software development success.* Information and Software Technology, 2002. **44**: p. 53-62.
10. Lederer, A.L. and J. Prasad, *Causes of Inaccurate Software Development Cost Estimates.* Journal of Systems and Software, 1995. **31**: p. 125-134.
11. van Genuchten, M., *Why is Software Late? An Empirical Study of Reasons for Delay in Software Development.* IEEE Transactions on Software Engineering, 1991. **17**(6).
12. Jiang, J. and G. Klein, *Software development risks to project effectiveness.* The Journal of Systems and Software, 2000. **52**: p. 3-10.
13. Ropponen, J. and K. Lyytinen, *Components of Software Development Risk: How to Address them? A project Manager Survey.* IEEE Transactions on Software Engineering, 2000. **26**(2).
14. Subramanian, G. and S. Breslawski, *An Empirical Analysis of Software Effort Estimate Alterations.* Journal of Systems and Software, 1995. **31**(2): p. 135-141.
15. Grimstad, S., M. Jørgensen, and K. Moløkken-Østvold, *Software Estimation Terminology - The Tower of Babel.* submitted to Information and Software Technology, 2005.

16. Gano, D., V. Lee, and V. Mitchell, *Apollo Root Cause Analysis - A New Way Of Thinking*. 1999: Apollonian Pubns.

17. Jørgensen, M., L. Moen, and N. Løvstad. *Combining Quantitative Software Development Cost Estimation Precision Data with Qualitative Data from Project Experience Reports at Ericsson Design Center in Norway*. in *Conference on Empirical Assessment in Software Engineering*. 2002. Keele, England: Keele University.

Paper IV:

Software Effort Estimation Terminology: The Tower of Babel

Stein Grimstad[1,2], Magne Jørgensen[1] and Kjetil Moløkken-Østvold[1,2]

[1]*Simula Research Laboratory*

[2]*University of Oslo*

Abstract

It is well documented that the software industry suffers from frequent cost overruns. A contributing factor is, we believe, the imprecise estimation terminology in use. A lack of clarity and precision in the use of estimation terms reduces the interpretability of estimation accuracy results, makes the communication of estimates difficult, and lowers the learning possibilities. This paper reports on a structured review of typical software effort estimation terminology in software engineering textbooks and software estimation research papers. The review provides evidence that the term 'effort estimate' is frequently used without sufficient clarification of its meaning, and that estimation accuracy is often evaluated without ensuring that the estimated and the actual effort are comparable. Guidelines are suggested on how to reduce this lack of clarity and precision in terminology.

1. Introduction

Software development effort estimates are the basis for project bidding, budgeting and planning. These are critical practices in the software industry, because poor budgeting and planning often has dramatic consequences. When budgets and plans are too pessimistic, business opportunities can be lost, while over-optimism may be followed by significant losses. The importance of accurate estimates is documented in a wide range of studies. For instance, the Standish Group [24] concludes that reliable estimation is among the top ten most important success factors in software projects. It is therefore unfortunate that, as indicated in a recent review of estimation surveys [41], there has been little improvement in software cost estimation accuracy over the last 20 years. We believe that one reason for this lack of improvement is the imprecise use of terminology for effort estimation. The following two case stories indicate that proper communication, interpretation and improvement of estimation accuracy measurements may be a problem when there is no precise use of terms related to estimation. This problem motivates the review and guidelines provided in this paper.

Case story 1: In 2003, two of the authors performed a survey on project estimation in Norwegian software companies [43]. The goal was to obtain an in-depth understanding of estimation practice and to examine factors that affect the accuracy of effort estimation. The basis for the measurement of estimation accuracy was a comparison of the actual use of effort with the estimated most likely effort provided in the planning stage of the project, i.e., the amount of effort the contractor believes that the project will require, regardless of the price to the customer or the budget. An interesting result was the observation that government projects had, on average, significantly higher deviations between estimated most likely efforts and actual efforts than private projects [42]. This observation made the headlines in Norway's largest morning newspaper. The day after the results were presented, the front page of the newspaper stated '*Yearly overruns of 6 billions [Norwegian Kroner] in governmental IT-projects*'[18]. The debate that followed was heated, and culminated in the research results being discussed in the Norwegian parliament. In particular, there were members of parliament who saw our results as evidence of a waste of government money on IT projects. However, our results did not say anything about the *customers'* budget overruns or losses. Neither had we studied the software providers' budget overruns or losses. What we *did* study was the overruns related to what the software providers believed was the *most likely effort* of a project. The newspaper article, which was the basis for the debate, did not point this out, i.e., we had failed to communicate the difference between overruns of most likely effort and overruns of budgeted costs. Budgeted costs typically include a risk buffer added to the most likely effort. The cost overrun we found through the survey was therefore probably much higher than the software organizations' and the customers' budget overrun. A consequence of the misinterpretation of the term 'cost estimate' was that the public discussion focused mainly on whether one should believe the high cost overrun number or not, and much less on how government projects could be better managed, i.e., on improvement of their role as software customers.

Case story 2: Some time ago, one of the authors was hired as a solution architect of a software project. It was a high risk project for a number of reasons: the functionality to be developed was complex, several stakeholders with conflicting goals were involved and a non-extendable deadline was set. The initial analysis suggested that the project would involve about 40 people and changes had to be made to five systems, all in operation. Our schedule and effort estimates suggested that we could probably deliver before the deadline, but with small margins. Not surprisingly, we ran into trouble during development and the changes to one of the systems were two weeks delayed. The changes to this system were on the project's critical path and the entire project was therefore two weeks delayed. Moving the deadline was, of course, unacceptable to the customer because this would ruin the announced launch. However, we did manage to deliver all functionality on time and on budget. We did so in the same way that many other software development teams do in similar situations; we reduced the amount of testing. The project went into operation, and luckily only minor failures occurred. How accurate were our estimates? From the outside, i.e., as would have been observed in most estimation surveys, we had only minor effort estimation error and no schedule overrun. In reality, however, the project would have had greater estimation error and a time overrun if the testing process had been completed as planned, i.e., with the promised level of quality. This case story shows that common ways of measuring effort estimation accuracy may give a misleading picture of the real estimation accuracy and hence a misleading picture of the need for improvement in the process of estimation and project management.

We present related work in Section 2 of this paper. Section 3 further elaborates on the consequences of imprecise use of effort estimation terminology. The consequences are illustrated by estimation error analyses of software projects in a Norwegian software development organization. In Section 4 we review the actual use of effort estimation terminology in popular software engineering textbooks and software estimation research papers. Based on the related work in Section 2, the discussion in Section 3 and the review in Section 4 we provide, in Section 5, recommendations aimed at improving the use of software effort estimation terminology, thus enabling improvement in the process of estimation. Section 6 concludes.

2. Related work

The problems of imprecise terminology for software cost estimation have been addressed by several software engineering researchers. Table 1 lists a number of research papers and textbooks that suggest there may be problems caused by the imprecise use of software estimation terminology. However, we have been unable to find any study with the same goal as this paper, i.e., to documenting the importance and the severity of the terminology problem through a structured review of popular software engineering textbooks and a representative set of estimation research papers.

Table 1 Related Work

Author	Examples of Estimation Terminology Problems Addressed
Kitchenham [33]	Kitchenham recommends that before you improve estimation processes, you should make sure that you do not have a management problem. She identifies lack of understanding of the probabilistic nature of estimates; confusing plans, costs and estimates; and not giving sufficient time to the estimation process as specific management problems.
DeMarco and Lister [14]	The authors argue that schedule flaws can occur when no distinction is made between the most optimistic estimate (that with virtually no probability of success), the goal (that which the project aims for), the estimate (the most likely outcome) and the schedule (what the project commits to).
DeMarco [13]	DeMarco points out that an estimate is a prediction based on a probabilistic assessment, and that an estimate should be the most likely value accompanied by upper and lower bounds. He says lack of experience is one of the important reasons for poor estimation and proposes that a separate metrics group should be responsible for data collection and estimation.
Boehm and Fairley [7]	Boehm and Fairly state two important points about software estimation: 1) It is best to understand the background of an estimate before using it and, 2) It is best to orientate the estimation approach to the use that is going to be made of the estimate.
Edwards and Moores [15]	The authors show that estimates can be a rough guide to the cost of a project as well as applying numbers to the detailed project plan. These meanings of estimates are different with respect to uncertainty, usage and motivation. They argue that the lack of clear distinction between these two types of estimates is why estimation tools are not commonly used in the industry.
Coombs [11]	Coombs explains that mixing price, cost and realistic estimates together with reduced functionality gives a false impression of estimation accuracy. He claims that this happens because most projects are underestimated to begin with and the only option left for the project managers is to axe the requirements and use both the contingency allowance and profit in order to meet the budget.
Jørgensen [25]	The author explains, through comparison with vacation cost estimation and by an industrial case study, why cost estimates with similar accuracy can hide huge differences in estimation performance. The paper exemplifies the conflicting goals of different types of estimates; 'most-likely software development cost', 'risk-minded planned development cost' and 'cost-reducing planned development costs'.

3. The Importance of Precise Effort Estimation Terminology

Obviously, the use of more precise or standardized terminology is not sufficient to solve most of the problems inherent in effort estimation. However, we believe that a *necessary* condition for sustainable improvement is the precise use of important terms, because lack of precision leads easily to the following:

- A mix of processes with different purposes, e.g., a mix of processes with a focus on realism (estimation of most likely effort), a focus on efficient development work (estimation of planned effort), a focus on avoidance of budget overrun (decisions on budgeted effort), and a focus on winning a bid (estimation of price-to-win). The lack of separation between these processes have been found to reduce the realism of estimation of most likely effort [10, 12, 32].
- Comparison of estimation error of different projects when they are not really comparable. For example, one project may have based their estimation error measurement on the difference between planned effort and actual effort, while another project may have based theirs on the difference between most likely effort and actual effort.
- Survey results that are difficult to interpret. If you want to evaluate your own estimation performance against those presented in an estimation survey, this is virtually impossible if the survey results are based on an unknown mixture of estimates of different types and different sizes of contingency buffers.

The consequences are improper evaluation, comparison and reporting of effort estimation performance, including a lower ability to learn from experience [26, 30].

Recent observations of software projects in a Norwegian software development organization further illustrate these consequences. Over a period of two years we logged estimation information of several software projects in that company as part of a study on reasons for estimation errors (a subset of the data is presented in [29]). As a part of the logging we requested that the person responsible for the estimation documented the estimate of ‘most likely effort’. An analysis of the description of how the estimate of ‘most likely effort’ was derived showed, however, a wide variety of interpretations. In most cases, the estimate was *not* of most likely effort. Instead, the effort estimate was typically described as most likely effort plus a risk buffer of varying size, i.e., it was interpreted as the planned or the budgeted effort, or was sometimes described as the effort (derived from the price) agreed with the customer. A consequence of the imprecise use of ‘effort estimate’ in the studied organization was that it was difficult to compare and evaluate the estimation accuracy of different projects. We compared the subset of projects that we assessed to contain effort estimates of ‘most likely effort’ with those we assessed to contain effort estimates of a type used as a basis for price to customer or planned effort, i.e., where a risk buffer typically was added to the most likely effort. The remaining projects were omitted from the comparison because it was difficult to classify the types of estimate used. The estimates of most likely effort had, on average, an effort overrun of 11%, while the estimates including a risk buffer on average used 8% less effort than estimated. From the description of the estimation process it seems that a typical risk buffer was 10-20% of most likely effort. When removing the specified risk buffer from the estimates that contained them, we found that they had, on average, almost the same estimation accuracy (about 10% overrun) as the estimates described as ‘most likely effort’. Finding the average estimation accuracy of all projects, without adjustments, would be like adding ‘apples and oranges’.

In the studied organization we also found it necessary to adjust the actual effort for the decreases and increases in delivered functionality to enable a proper interpretation. Several of the projects had increases or decreases in functionality of more than 10%. In most cases this adjustment led to better estimation accuracy; many estimates of most likely effort were accurate, but looked inaccurate because of added or removed functionality. For example, a project went from 50% effort overrun to 10% overrun when we adjusted for the increase in functionality. Again, without this adjustment a comparison of estimation accuracy would give a misleading picture of the estimation ability in many of the projects.

We have logged similar estimation information for other organizations and believe that this is a common pattern [28]. The results are further supported by the survey described in the first section of this paper [43], which indicates, for instance, that the separation between price and most likely effort is blurred in many organizations.

Lack of precise terminology for estimation may hinder the improvement of estimation processes. It is, for example, difficult to determine whether estimation errors are caused by poor estimation ability, poor risk analysis, poor project management, or something else, when a clear estimation terminology is absent.

4. A Review of Textbooks and Previous Research

4.1. Design of Review Process

In order to examine possible reasons for the lack of precise estimation terminology in the software industry, we reviewed the actual use of estimation terminology in a representative set of estimation research papers and the most popular software engineering textbooks. The motivation for this selection is that we believe these are the publications that have the most influence on the estimation terminology used by software professionals. The review focuses on the following two questions derived from the software cost estimation terminology problems discussed in the previous sections:

- *Q1: Is the term 'effort estimate' precisely defined?*
- *Q2: When evaluating estimation accuracy, are the estimates and the actual efforts comparable?*

In order to conduct the review in a fair and auditable manner, the review design is based on the guidelines for systematic reviews proposed by Kitchenham [34]. Note that the review design deviates somewhat from these guidelines. The main reason for the deviations is that we aim at describing 'typical practices', not 'best practices' or 'all practices', regarding use of estimation terminology. Section 4.2 describes how the reviewed material was selected, while Section 4.3 explains how the review questions (Q1 and Q2) were assessed for the reviewed material. The results of the review are presented in Section 4.4. The validity of the review is discussed in Section 4.5, and Section 4.6 provides a general discussion of the results.

4.2. Selection of Reviewed Material

We used different approaches to select textbooks and research papers. As we aimed at selecting textbooks that address software cost estimation and are among those most often read by software professionals, we targeted books used by universities in lecturing and books that software professionals read. For research papers, we consider precise estimation terminology to be most important for papers that report on estimation accuracy. Therefore, we aimed at selecting a representative sample of research papers in journals and conferences that report on software cost estimation accuracy.

Three different information sources were used to identify the literature. Lecture books were found by using Google to search the internet using the search string 'software engineering course books'. We then manually investigated the first 100 URLs that appeared relevant, and counted the frequency of each textbook used in university courses on software engineering. The four most popular textbooks were included in the review.

Books that software professionals are most likely to read were selected by using Amazon's list of the top 100 bestselling computer science books. Books were extracted from the bestselling list by browsing the list manually. For a book to be included in the review, we had to regard it as likely that the book addresses software cost estimation in particular. These judgments were based on the title and the abstract of the book. The assessments were done by two of the authors, independently of each other. When there were disagreements, the book were included. If the same author had multiple books that appeared to meet the selection criteria, only the most recent were selected. An examination of the books revealed that three of the books did not include any sections on software cost estimation. These three books were excluded from the review.

Research papers were selected from the BESTweb library (available at www.simula.no/BESTweb). BESTweb is an online library of estimation papers that claims to include nearly all journal papers and a large proportion of the conference papers on software cost estimation. The journal papers in BESTweb were selected by manually scanning potentially relevant journals, while conference papers were identified by a comprehensive search in the INSPEC-library. A full description of the BESTweb library will appear in a forthcoming paper by one of the authors. At the time of the review, the BESTweb library contained 963 estimation relevant articles. We selected papers to include in the review by reading all the abstracts, and then the full versions of all the papers that appeared to meet all inclusion criteria:

- Deal with estimates of software development effort, schedule, budget or cost or with project success/failure/performance.
- Report on empirical collected estimates from real projects (not experiments or student projects).
- Report on estimates made up-front. This excludes, for example, all history-based evaluations of formal estimation models.
- Report on estimation accuracy.

- The paper was the most recent paper by the main author that met the above criteria. This criterion applied to several of the authors (for instance, Lederer and Prasad had several older papers excluded).

This selection procedure has some limitations, e.g., it is not a review of all relevant estimation papers and the initial selection of papers by title and abstract only may lead to exclusion of papers that otherwise meet the inclusion criteria. For the purpose of documenting typical estimation terminology practice, however, the selection process is, in our opinion, acceptable.

The searches for lecture books and bestselling books were conducted on the 23rd of February 2005. Amazon's best selling lists report those books that currently have the most sales. Similarly, most of the web pages of university courses in software engineering were recently updated. The search for research papers was conducted on the 19th of February. BESTweb covers articles published up to April 2004. The selection of reviewed lecture books and research articles were done by one of the authors, while two of the authors selected bestselling books.

4.3. Assessment of the Research Questions (Data Extraction)

The first review question (Q1: Is the term ‘effort estimate’ precisely defined?) is evaluated to have been answered satisfactorily if there is a definition of 'effort estimate' in the reviewed material that clarifies whether the intended meaning of the term is an estimate of 'most likely effort', 'budgeted effort', 'price' or something else. The question is also deemed to have been answered satisfactorily if, even if there is no explicit definition, the terminology in use makes a clear and consistent distinction between estimates made for different purposes.

The second review question (Q2: When evaluating estimation accuracy, are the estimates and the actual efforts comparable?) is evaluated to have been answered satisfactorily if the comparability of estimated effort and actual effort is discussed, or actions to ensure comparability are taken in situations when there may be significant differences in functionality or quality between the estimated and the actual solution. For example, comparable values can be secured either through adjustments of actual effort or removal of projects in cases where estimated and actual effort is not comparable.

Assessment of the research questions was conducted by two of the authors, independently of each other. When a question was not a topic addressed in a textbook or a paper, we used the value ‘na’ (not addressed). There were only minor disagreements to be resolved. Disagreements were resolved by examining each controversial issue separately. We reached an agreement in all the cases.

4.4. Review Results

The reviewed material and our evaluations of their use of estimation terminology are presented in Table 2 (textbooks) and Table 3 (research papers). In the tables, the first column identifies the reviewed publication, the second and third columns report the type of publication and the fourth and fifth columns include our answers of the review questions (Q1 and Q2).

The results of the review show that the term 'effort estimate' is rarely used in a consistent manner (Q1) neither in textbooks nor research papers. Only one [39] out of the eight books and two [27, 35] out of the 23 research papers use estimation related terminology in a way we find satisfying according to the criteria described in Section 4.3. However, a few of the texts partly comply to the criteria. For instance, in [22] there is a distinction between estimates for pricing and estimates for planning, but no distinction between 'planned effort' and 'most likely effort'. For the majority of the reviewed material we were unable to tell whether the term 'effort estimate' referred to an estimate of 'most likely effort', 'budgeted effort', 'price' or something else.

Whether estimated effort is comparable to actual effort when evaluating estimation accuracy (Q2) is a topic in two of the textbooks [21, 39]. They both include a brief discussion of problems related to estimation accuracy evaluation, but neither of them suggest guidelines or provide any example of how the problems can be solved in practice. In the research papers, Q2 is addressed in nine of the papers [1, 4-6, 22, 35, 40, 47, 51]. They handle the incomparability in a somewhat different manners: Some studies discuss the consequence of incomparability or assess it to be ignorable/not relevant [1, 4, 5, 40, 47], some studies remove data points [22, 35], while one study avoids to calculate estimation accuracy at all due to comparison problems [51]. Only one of the studies attempts to adjust the actual effort to be comparable to the original estimate [6].

Table 2 Textbooks

Publication	Lecture book	Bestselling book	Q1	Q2
Brooks [9]	Y	Y	N	NA
Heldman [21]	N	Y	N	Y
Larman [36]	Y	N	N	NA
McConnell [39]	N	Y	Y	Y
Pressman [44]	Y	Y	N	NA
Schwaber and Beedle [46]	N	Y	N	NA
Sommerville [48]	Y	N	N	NA
Sponsky [49]	N	Y	N	NA

4.5. Threats to Validity

We assess the major threats to validity to be related to: 1) Biased selection of textbooks or research papers, 2) Biased review of the textbooks or research papers.

Biased Selection: The bestselling list at Amazon and the first 100 hits on Internet is a sample based on current popularity and not, for example, quality. This type of sample was intended as a means to review typical practice, but it does give a poor picture of 'best practice' among textbooks. We acknowledge that there are textbooks with rather precise estimation terminology. Another limitation to the review is that it only includes books and articles that address software cost estimation. This means that related material, such as general project management and forecasting literature, where a more precise terminology might be present, was not reviewed. However, our impression is that such literature is not much read by most software professionals.

Biased Review: The review was conducted by two reviewers, independently of each other. However, these reviewers are from the same research group and so they are not totally independent, and other reviewers may answer the questions differently. In addition, some of the reviews may be highly subjective. Despite these potential sources of bias, we believe that the main conclusion is quite robust: most textbooks and research papers on estimation are not based on a precise use of estimation terminology.

Table 3 Research Papers

Publication	Conference paper	Journal paper	Q1	Q2
Abdel-Hamid et. al. [1]	N	Y	N	Y
Barki et. al. [3]	N	Y	N	N
Barry et. al. [4]	N	Y	N	Y
Bergeron and St-Arnaud [5]	N	Y	N	Y
Berry and Schoenborn [6]	Y	N	N	Y
Bootsma [8]	Y	N	N	N
Fleck [16]	N	Y	N	N
Gray et. al. [17]	Y	N	N	N
Haynes and Henderson-Sellers [19]	N	Y	N	N
Heemstra and Kusters [20]	N	Y	N	N
Hill et. al. [22]	N	Y	N	Y
Jenkins et. al. [23]	N	Y	N	N
Jørgensen [27]	N	Y	Y	N
Kamatar and Hayes [31]	N	Y	N	N
Kitchenham et. al. [35]	N	Y	Y	Y
Lederer and Prasad [37]	N	Y	N	N
Lind and Sulek [38]	N	Y	N	N
Mizuno et. al. [40]	Y	N	N	Y
Moløkken and Jørgensen [41]	Y	N	N	N
Ropponen and Lyytinen [45]	N	Y	N	N
Shepperd and Cartwright [47]	N	Y	N	Y
Subramanian and Breslawski [50]	N	Y	N	N
Taff et. al. [51]	N	Y	N	Y

4.6. Discussion of Results

The previous sections suggest that an important obstacle for estimation improvement is imprecise estimation terminology and that a reason for the lack of precise use of estimation terminology is the lack of precise terminology in software textbooks and research papers. However, it might also be that the direction of cause and effect is reversed, as well. It is difficult to survey estimation practice and write good estimation guidelines when important estimation terms are vague and used inconsistently by software professionals. Consequently, attempts to improve the use of estimation terminology should be made concurrently in both industry practice and the writing of textbooks and research papers.

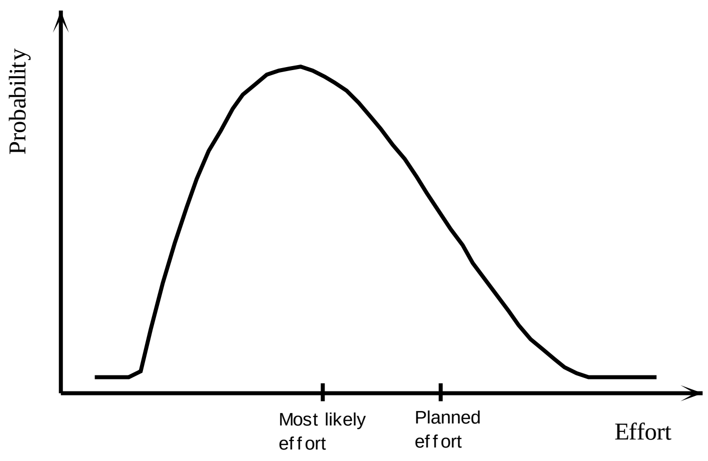
There may be different motivations for improving the precision of estimate terminology. Software organizations may wish to improve their use of estimation terminology to avoid misunderstandings, to increase the realism in the estimates, and to facilitate learning from experience. Software researchers may wish to develop precise terminology to increase the validity of their research results, (e.g., when comparing two formal estimation models), and to suggest better estimation guidelines for software professionals.

Since several researchers have pointed out the importance of a precise software estimation terminology, e.g., DeMarco [13] as early as in 1982, it is somewhat surprising that software estimation has been conducted out for so many years without greater attention being paid to the use of precise terminology. There are a number of possible reasons for this lack of progress:

- Authors of the estimation literature seem to take a "deterministic" (estimates as one single effort value) instead of a "probabilistic" (estimates as a combination of effort value and probability) view on effort estimation. A probabilistic view means here that 'most likely effort', 'planned effort', 'budgeted effort', etc., are values (with different probabilities of being exceeded by actual effort) on an effort probability distribution. Figure 1 illustrates how estimates of most likely effort (the effort with the highest probability) and planned effort (typically, most likely effort plus a contingency allowance) are values on a probability distribution of effort. Without a probabilistic basis of effort estimation terminology a separation of most likely, planned, and budgeted effort may be difficult to describe. The strong textbook focus on parametric cost estimation models, which typically deliver only a single effort value, may be one reason for the adoption of a deterministic view.
- Software organizations do not regard estimation as a separate activity, but as an integrated part of project planning, project pricing and project budgeting. As pointed out earlier, mixing processes may result in the mixing of terminology.

- Software organizations typically do not collect the data necessary to validate and adjust the actual effort to make it comparable with the estimated effort. Our experience is that most organizations have an unformed view on how to assess estimation accuracy measurements and do not allocate any resources to the in-depth analysis of estimation accuracy data across projects [30].

Figure 1 Example of an Effort Probability Distribution



5. Guidelines for Estimation Terminology

Our review motivates a change towards a more proper software cost estimation terminology among software professionals and researchers. Proper estimation terminology is a complex topic and it is beyond the scope of this paper to provide suggestions for a complete terminology. We propose, however, two simple guidelines, the following of which is, we believe, essential for improved software estimation processes. The guidelines are aimed at all users of software cost estimation terminology, including authors, practitioners, researchers and reviewers. The guidelines are based on our own experience and recommendations made in the text books and papers summarized in Table 1.

Guideline 1: Do not mix estimation of most likely effort with planning, budgeting or pricing.

Implications of the guideline for researchers and authors of textbooks:

- Different terms should be used for different concepts. In particular, a distinction should be made between estimated ‘most likely effort’, ‘planned effort’ and ‘budgeted effort’.
- When conducting surveys or logging estimation information, it must not be assumed that the terminology used is understood, even if it is defined precisely. In-depth studies and triangulation may be needed to ensure that all the data are based on the same understanding of the estimation terminology used.

Implications of the guideline for practitioners:

- Different terms should be used for different concepts. In particular, a distinction should be made between estimated ‘most likely effort’, ‘planned effort’ and ‘budgeted effort’.
- The estimation of most likely effort should be performed as an independent activity and separated from planning, budgeting and pricing. People in charge of bidding should, for example, not be in charge of the estimation of most likely effort, to ensure that pricing and realism are not mixed. Planning tools should not be used as estimation tools, or, at least, used with great care to avoid a mixing of concerns.

Guideline 2: When assessing estimation accuracy, make sure that the estimated and the actual effort are comparable.

Implications of the guideline for researchers:

- The actual efforts should be adjusted so that they are comparable to the estimated effort with respect to technical and functional parameters before the estimation accuracy is calculated. If functional and quality requirements are not available, the project plan should be investigated and interviews should be used to identify changes in scope and/or quality. If estimates are of types other than most likely effort estimates, they should be transformed to most likely estimates before the accuracy is calculated.

- When estimates cannot be reliably transformed to values that are comparable to the actual result, great care should be taken when using these results, or the projects for which such transformation cannot be performed should be removed from the data set.

Implications of the guideline for practitioners:

- The scope and other assumptions of the estimate of most likely effort should be recorded. The version of the requirement specification, and other documents that the estimate of most likely effort is based on, should be specified.
- Deviation from estimated scope, quality, and development process should be recorded.

The first of the case stories in Section 1 presents an example of how violation of Guideline 1 (estimates of most likely effort not clearly separated from budgeted effort) resulted in a public debate where a research report on overruns of most likely estimates was mistakenly used as evidence of governmental waste of money. An example of the second guideline's importance is presented in the case study described in Section 3. This case study shows how violation of Guideline 2, (no adjustment of estimation accuracy when the estimated and the actual solution differs), would lead to unfair evaluation of estimation ability. More comprehensive estimation terminology guidelines can be found in [2, 26, 33].

6. Summary

Effort and schedule overruns are serious problems in the software industry. In this paper we argue that the lack of a precise software effort estimation terminology is an important obstacle for the improvement of estimation accuracy. We reviewed the currently most popular software textbooks and a representative set of software estimation research papers and found systematic shortcomings in use of estimation terminology. For example, estimates of most likely effort are frequently mixed with planned effort, budgets and price. In addition, effort estimation accuracy is frequently measured without adjustments being made for differences in the scope and/or quality assumed when estimating the effort and the system actually implemented.

In order to improve effort estimation accuracy, a more precise terminology for software effort estimation is needed. We provide two simple guidelines for this purpose: 1) Do not mix estimation of most likely effort with planning, budgeting or pricing, and 2) When assessing estimation accuracy, ensure that the estimate and the actual effort are comparable. Although these guidelines are not innovative and might seem obvious, they are nevertheless worth stressing. As this review points out, they are frequently violated.

Acknowledgement

Thanks to Professor Barbara Kitchenham, Professor Ray Welland and the anonymous reviewers for useful comments and suggestions. Also, thanks to the Norwegian Research Council who sponsored this work through the SPIKE and INCO projects.

References

[1] T. K. Abdel-Hamid, "Adapting, correcting, and perfecting software estimates: a maintenance metaphor," *IEEE Computer*, vol. 26, no. 3, pp. 20-29, 1993.

[2] J. S. Armstrong, "Standards and practices for forecasting," in *Principles of forecasting:A handbook for researchers and practitioners*. Boston: Kluwer Academic Publishers, 2001.

[3] H. Barki, S. Rivard, and J. Talbot, "An integrative contingency model of software project risk management," *Journal of Management Information Systems*, vol. 17, no. 4, pp. 37-69, 2001.

[4] E. J. Barry, T. Mukhopadhyay, and S. A. Slaughter, "Software project duration and effort: an empirical study," *Information Technology & Management*, vol. 3, no. 1-2, pp. 113-136, 2002.

[5] F. Bergeron and J. Y. St-Arnaud, "Estimation of information systems development efforts: a pilot study," *Information and Management*, vol. 22, no. 4, pp. 239-254, 1992.

[6] R. H. Berry and R. M. Schoenborn, "Estimating requirements for a large, software engineering project (experience with Ada COCOMO on SIDPERS-3)," *proc. TRI-Ada '92*, pp. 375-383, 1992.

[7] B. Boehm and R. Fairley, "Software estimation perspectives," *IEEE Software*, vol. 17, no. 6, pp. 22-26, 2000.

[8] F. Bootsma, "How to obtain accurate estimates in a real-time environment using full function points," *proc. IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, pp. 105-112, 2000.

[9] F. Brooks, *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*: Addison-Wesley Professional, 1995.

[10] R. Buehler, D. Griffin, and H. MacDonald, "The role of motivated reasoning in optimistic time predictions," *Personality and Social Psychology Bulletin*, vol. 23, no. 3, pp. 238-247, 1997.

[11] P. Coombs, *IT Project Estimation - A Practical Guide to the Costing of Software*. Cambridge: Cambridge University Press, 2003.

[12] R. A. Cosier and G. L. Rose, "Cognitive conflict and goal conflict effects on task performance," *Organizational Behaviour and Human Performance*, vol. 19, no. 2, pp. 378-391, 1977.

[13] T. DeMarco, *Controlling Software Projects*. Upper Saddle River: Prentice Hall PTR, 1982.

[14] T. DeMarco and T. Lister, *Waltzing with bears : managing risk on software projects*. New York: Dorset House, 2003.

[15] J. S. Edwards and T. T. Moores, "A conflict between the use of estimating and planning tools in the management of information systems," *European Journal of Information Systems*, vol. 3, no. 2, pp. 139-147, 1994.

[16] R. A. Fleck, Jr., "Managing programmer resources in a maintenance environment with function points," *Industrial Management + Data Systems*, vol. 98, no. 2, pp. 63-70, 1998.

[17] A. Gray, S. MacDonnell, and M. Shepperd, "Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment," *proc. International Software Metrics Symposium*, pp. 216-227, 1999.

[18] G. M. Haugnes, "Offentlig IT-sprekk for 6 mrd. hvert år," in *Aftenposten*, j ed. Oslo, 2004.

[19] P. Haynes and B. Henderson-Sellers, "Cost estimation of OO projects: empirical observations, practical applications," *American Programmer*, vol. 9, no. 7, pp. 35-41, 1996.

[20] F. J. Heemstra and R. J. Kusters, "Function point analysis: Evaluation of a software cost estimation model," *European Journal of Information Systems*, vol. 1, no. 4, pp. 223-237, 1991.

[21] K. Heldman, *PMP: Project Management Professional Study Guide*. Alamenda: SYBEX Inc., 2002.

[22] J. Hill, L. C. Thomas, and D. E. Allen, "Experts' estimates of task durations in software development projects," *International Journal of Project Management*, vol. 18, no. 1, pp. 13-21, 2000.

[23] 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.

[24] J. Johnson, K. Boucher, K. Connors, and J. Robinson, "The Criteria for Success - Industry Trend or Event," in *Software Magazine*, vol. February, 2001.

[25] M. Jørgensen, "How much does a vacation cost?," *Software Engineering Notes*, vol. 28, no. 6, p. 30, 2003.

[26] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37-60, 2004.

[27] M. Jørgensen, "Realism in assessment of effort estimation uncertainty: It matters how you ask," *IEEE Transactions on Software Engineering*, vol. 30, no. 4, pp. 209-217, 2004.

[28] M. Jørgensen and D. I. K. Sjøberg, "Impact of effort estimates on software project work," *Information and Software Technology*, vol. 43, no. 15, pp. 939-948, 2001.

[29] M. Jørgensen and K. Moløkken-Østvold, "Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 993-1007, 2004.

[30] M. Jørgensen, L. Moen, and N. Løvstad, "Combining Quantitative Software Development Cost Estimation Precision Data with Qualitative Data from Project Experience Reports at Ericsson Design Center in Norway," *proc. Conference on Empirical Assessment in Software Engineering*, 2002.

[31] J. Kamatar and W. Hayes, "An experience report on the personal software process," *IEEE Software*, vol. 17, no. 6, pp. 85-89, 2000.

[32] P. G. W. Keen, "Information systems and organizational change," *Social Impacts of Computing*, vol. 24, no. 1, pp. 24-33, 1981.

[33] B. Kitchenham, *Software Metrics: Measurement for Software Process Improvement*: Blackwell Publishers, 1996.

[34] B. Kitchenham, "Procedures for Performing Systematic Reviews," Keele University, Keele, Technical Report 2004.

[35] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan, "An empirical study of maintenance and development estimation accuracy," *Journal of Systems and Software*, vol. 64, no. 1, pp. 57-77, 2002.

[36] C. Larman, *Applying UML and Patterns: an introduction to object-oriented analysis and design and iterative development*, 3rd ed. Upper Saddle River: Pearson Education, Inc., 2005.

[37] A. L. Lederer and J. Prasad, "Causes of inaccurate software development cost estimates," *Journal of Systems and Software*, vol. 31, no. 2, pp. 125-134, 1995.

[38] M. R. Lind and J. M. Sulek, "Undersizing software systems: third versus fourth generation software development," *European Journal of Information Systems*, vol. 7, no. 4, pp. 261-268, 1998.

[39] S. McConnell, *Rapid Development*. Redmond: Microsoft Press, 1996.

[40] O. Mizuno, T. Kikuno, K. Inagaki, Y. Takagi, and K. Sakamoto, "Statistical analysis of deviation of actual cost from estimated cost using actual project data," *Information and Software Technology*, vol. 42, no. 7, pp. 465-473, 2000.

[41] K. Moløkken and M. Jørgensen, "A review of software surveys on software effort estimation," *proc. International Symposium on Empirical Software Engineering*, pp. 223-230, 2003.

[42] K. Moløkken, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove, "Project Estimation in the Norwegian Software Industry – A Summary," Simula, Technical Report 2004.

[43] K. Moløkken-Østvold, M. Jørgensen, S. Tanilkan, H. Gallis, A. Lien, and S. Hove, "A Survey on Software Estimation in the Norwegian Industry," *proc. Metrics '04*, pp. 208-219, 2004.

[44] R. S. Pressman, *Software engineering - a practitioner's approach*, 6. ed: McGraw-Hill, 2005.

[45] J. Ropponen and K. Lyytinen, "Can software risk management improve system development: an exploratory study," *European Journal of Information Systems*, vol. 6, no. 1, pp. 41-50, 1997.

[46] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Upper Saddle River: Prentice Hall, 2002.

[47] M. Shepperd and M. Cartwright, "Predicting with sparse data," *proc. International Software Metrics Symposium*, pp. 28-39, 2001.

[48] Sommerville, *Software Engineering*, 7. ed: Addison-Wesley, 2004.

[49] J. Sponsky, *Joel on software: and on diverse and occasionally related matters that will prove of interest to software developers, designers, and managers, and to those who, whether by good fortune or ill luck, work with them in some capacity*. Berkeley: Apress, 2004.

[50] G. H. Subramanian and S. Breslawski, "An empirical analysis of software effort estimate alterations," *Journal of Systems and Software*, vol. 31, no. 2, pp. 135-141, 1995.

[51] L. M. Taff, J. W. Borchering, and J. W. R. Hudgins, "Estimeetings: development estimates and a front-end process for a large project," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 839-849, 1991.

Paper V:

A Framework for the Analysis of Software Cost Estimation Accuracy

Stein Grimstad[1,2] and Magne Jørgensen[1]

[1]*Simula Research Laboratory*

[2]*University of Oslo*

Abstract

Many software companies track and analyze project performance by measuring cost estimation accuracy. A high estimation error is frequently interpreted as poor estimation skills. This is not necessarily a correct interpretation. High estimation error can also be a result of other factors, such as high estimation complexity and insufficient cost control of the project. Through a real-life example we illustrate how the lack of proper estimation error analysis technique can bias analyses of cost estimation accuracy and lead to wrong conclusions. Further, we examine a selection of cost estimation studies, and show that they frequently do not take the necessary actions to ensure meaningful interpretations of estimation error data. Motivated by these results, we propose a general framework that, we believe, will improve analyses of software cost estimation error.

1. Introduction

Software cost estimation is an essential part of most software development [7, 8, 22]. Unfortunately, software development cost estimation is difficult and inaccurate. In spite of the availability of many estimation methods and guidelines, e.g., [5, 12, 20], there is still a need for improvement. One means of reducing cost estimation error is through analysis of cost estimation error, e.g., through the identification of estimation processes that lead to more accurate estimates. Cost estimation error is relatively easy to measure. Unfortunately, as shown in this paper, the measured data can be hard to interpret properly. If we do not know what we measure, there may be little to learn from estimation error analysis. By contrast, proper measurement and analyses of estimation error may have the following beneficial results:

- identification of estimation processes resulting in systematic lowering of estimation errors
- improved evaluation and training of people responsible for estimation
- identification of (controllable and not controllable) factors that lead to estimation error, which will enable improved risk management

The goal of this paper is to show that commonly used estimation error analysis processes can lead to wrong conclusions, to document an insufficient focus on removing the effect from non-studied variable in estimation error analyses, and to propose a framework for improved analysis of software cost estimation error.

The paper is organized as follows: Section 2 describes a real-world example where a straight-forward analysis of estimation error led to misleading conclusions and our attempt to improve the analysis. Section 3 examines estimation error analyses in software cost estimation studies. Section 4 presents a framework for analysis of estimation error, and Section 5 summarizes.

2. A Real-World Example

2.1. The Company

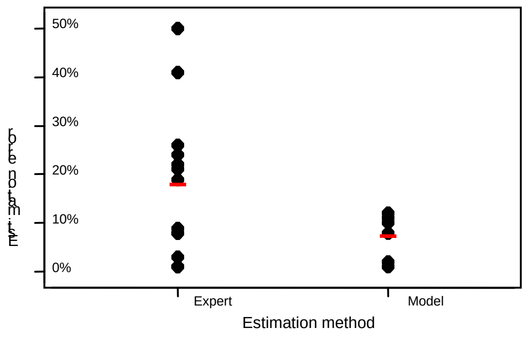
Software development company X had estimated the cost of their projects in two ways; some had been estimated by applying a self-developed estimation model and some by expert estimation. The estimation model was based on counting and classifying program elements, e.g., GUI-elements and program components. Each program element was classified according to size (very small, small, medium, large, very large) and technical complexity (very low, low, medium, high, very high). The estimation model suggested the required work-hours for the programming and unit test of each program element based on input of size and complexity. The estimation model had effort values calibrated to different types of development project. Since the classification of size and complexity is expert judgment-based, the estimation model may be characterized as a combination of model and expert judgment. Despite the availability of this estimation model, most projects were estimated by unsupported expert judgment-based estimation. A typical expert estimation-based process was to break the project work into activities and to estimate the effort of each activity, i.e., bottom-up, expert judgment-based estimation.

After some time, the company wanted to know whether the estimation performance was better with the model or with the expert estimation. The goal of the estimation error analysis was to compare the performance of the self-developed estimation model with that of expert estimation.

2.2 The Straight-Forward Analysis

To compare the performance of the model and the expert judgment-based effort estimates, the company collected data from 19 completed projects. The company found that the mean estimation error was 18% for the expert judgment-based estimates, yet only 7% for the model-based ones. The difference was statistically significant (t-test of difference in mean values gave p=0.04). The estimation error data are displayed in Figure 1, which clearly suggests that more use of model-based estimation would lead to lower estimation error.

Figure 1: Estimation Error Data
(means are indicated by lines)



This straight-forward analysis of real-world estimation error data tells a convincing story about the benefits of using the estimation model. Unfortunately, as we discovered as part of a larger study of the organization's estimation performance, this convincing story is probably flawed.

2.3 Our Attempt to Improve the Analysis

The analysis in Section 2.2 is what could have happened if a typical data collection and analysis process had been followed. This section describes how we conducted the data collection and the analysis and how this changes the conclusion.

Our attempt to improve the estimation error analysis started with an examination of the implications of the goal of the analysis. The general *goal* of the analysis was to get more knowledge about the estimation ability (performance) of two different categories of estimation method: the estimation model and the expert estimation. Clarifying the goal, we found that the estimation accuracy of the smallest projects, here defined as projects with less than 100 work-hours, was not essential for the organization and they should, consequently, be excluded from the analysis. Similarly, very large projects were excluded from the analysis for two reasons: (i) they were considered to be substantially different from the other projects with respect to estimation complexity, and (ii) there were too few to be analyzed by statistical means. The factor to be analyzed was, consequently, the relative estimation performance of two estimation methods for medium-size (between 100 and 5000 work-hours) projects.

From previous experience we knew that there might be systematic method use biases with respect to when formal estimation models and expert estimation are used. In particular, we anticipated that:

- Early estimates based on limited information would usually be based on expert estimation, because experts are better able to cope with incomplete data and non-standard data formats than the model.
- The calibration of the formal estimation model required data from similar projects. This means that expert estimation may have been the only possible option for projects when the company had few similar projects.

In addition, we decided to collect data related to the following factors that could potentially bias the comparison of the estimation error of the model and the expert judgment:

- Difference between estimated and actual functionality.
- Unusual reasons for estimation error.
- Differences in the project's priorities on cost control.

Other differences, e.g., difference in the level of estimation skill between users of estimation models and expert estimation, were assumed to be small or unlikely to be more systematically associated with one estimation method than the other. We assumed, based on information about the experience of the estimators, that the estimation model was applied properly. As can be seen, the identification of factors of interest did require knowledge about the studied process and the organization. Notice also that the estimation model we compare with expert estimation is a rather simple two-parameter (size and technical complexity) model. More sophisticated formal estimation models may, of course, perform better and this is no general evaluation of models vs. experts.

The second author of this paper was fortunate in that he was able to influence the amount and quality of the estimation related data that were logged. Estimation-related information was collected and logged by the estimators as soon as a project estimate was completed and immediately after the project was completed.

We instructed the estimators to interpret “effort estimate” as “most likely effort” when logging the estimation data. In addition, we required that the estimation process should be described briefly for each estimate. This description of the estimation process enabled us to check whether the estimate was a most likely effort estimate and to check the actual use of the estimation methods. We used MRE (=|actual effort – estimated effort|/actual effort) and RE (=(actual effort – estimated effort)/actual effort) as measures of estimation error.

We decided to use a combination of exclusion, adjustment and grouping to isolate the estimation performance factor:

1) Exclude projects with very unusual reasons for estimation errors, that is where the dominant reason for high estimation accuracy or error was clearly not related to the performance of the estimation method or other studied factors.

2) Adjust the actual effort for large differences, i.e., more than 10%, between estimated and actual functionality. This was done to ensure that the actual effort was comparable to the estimated effort with respect to technical and functional parameters before the estimation accuracy was calculated, as recommended in [10].

3) Compare estimation error collected in similar estimation situations, e.g., similar with respect to:

- Time of estimation: a) early estimate used as input to bidding, and, b) estimate used as input for planning.
- Cost control priority of the project: a) priority on cost control, b) priority on time control or quality.
- Estimation complexity: a) the estimator had experience from a similar project, b) the estimator did not have experience from a similar project.

Estimation data from 56 projects were collected. As many as 21 of these projects were either not started or never completed, and they were therefore excluded. This may have lead to some serious measurement selection bias if we aimed at measurement of the organization's estimation skill. We assessed, however, that the reasons for non-completions were not related to use of estimation method and that this therefore would not bias our comparison of the two estimation methods.

Among the 35 completed projects, 19 had a size larger than 100 work-hours. Among those 19 projects, 13 were based on expert judgment and six on the model. (These 19 projects are the same as those displayed in Figure 1).

In spite of our instruction of interpreting 'estimate' as 'most likely effort', we suspected, from the analysis of the estimation process descriptions, that there were variations in what the estimators meant by 'estimate'. Some of the estimators reported an estimate of most likely effort, as instructed, and others reported planned effort which included a small contingency buffer. Our analysis is based on the assumption that this will not bias the comparison. For future measurements we would recommend that the organization trained the estimators in consistent use of estimation terminology.

As shown in Section 2.2 a straight-forward analysis of the difference between the two estimation methods suggested that the estimation model led to significantly[*] more accurate estimates compared with that of the expert estimation (MRE of 0.07 vs MRE of 0.18). The following analysis shows that the result from the first analysis indeed is questionable and that other conclusions are better supported.

In the improved analysis we started with an exclusion of the outliers. We found only one obvious outlier. This project had a very high flexibility in product delivery and was removed from the analysis. The estimate of that project was better described as a 'budget' and the goal was to a large extent to deliver as much functionality as possible within the budgeted resource usage. Eighteen projects remained. This is a small number for estimation error analysis, but not an unusual number of observations available for software companies to analyze.

Next, we adjusted for differences between estimated and actual functionality. The difference between estimated and actual functionality was asserted and reported by the estimators at the end of each project. We found two projects with more than a 10% difference between estimated and actual functionality. We assumed that an X% increase in functionality was connected with an X% increase in effort and adjusted for this increase in the actual effort. This assumption of a linear relationship between functionality is only an approximation of the true relationship. As shown in [9], a statistically significant non-linear relationship between effort and size is not strongly supported by existing empirical data.

Then, we grouped and analyzed the differences with regard to time of estimation, project priority, and experience with similar projects. Results from the grouping are displayed in Table 1 and Table 2.

[*] Notice that statistical significance analysis of difference in mean values requires random allocation of treatment. It is therefore not a proper analysis for this and most other real-world software industry data set analyses.

Table 1 Grouped Analysis, Estimation Model

Factor	Value	#	Mean MRE	Mean RE
Time of estimation	Early Estimate	1	0.08	-0.02
	Planning Phase	5	0.07	0.08
Project priority	Cost	4	0.09	0.03
	Time or Quality	2	0.05	-0.02
Experience from similar projects	Yes	6	0.07	0.00
	No	0	-	-

Table 2 Grouped Analysis, Expert Estimates

Factor	Value	#	Mean MRE	Mean RE
Time of estimation	Early Estimate	6	0.15	0.05
	Planning Phase	6	0.14	0.10
Project priority	Cost	1	0.08	0.08
	Time or Quality	11	0.15	0.07
Experience from similar projects	Yes	7	0.10	0.01
	No	5	0.20	0.16

Then, we identified differences in frequency of use of the model and the expert judgment-based estimation methods, i.e., we compared the values in the # columns in Table 1 and Table 2, and found that expert estimates had a larger proportion of their estimates in the early phase, when the focus was on time or quality, and, when the estimators had no previous experience from similar projects. All these situations are connected with higher estimation complexity and higher estimation error, as can be seen from the mean MRE and RE-values in the tables. We, consequently, started to suspect that the higher estimation error of expert estimates could be explained by a more frequent use in higher estimation complexity situations and not by lower estimation performance of the method.

This alternative explanation would be supported if the difference in estimation error disappeared if we compared projects with similar estimation complexity characteristics only. Optimally, we should have compared for each combination of factor values, e.g., for the combination of “early estimate”, “cost priority” and “experience from similar projects”. The low number of data points did not enable such analyses. Instead, we had to rely on a mix of individual analysis of each factor and informal assessment of the contribution of the other factors. Table 1 and Table 2 display data for the purpose of the comparison. Notice that several of the mean values are based on quite few observations and should be interpreted especially carefully.

Table 1 and Table 2 suggest, as we understand the data, that there is at least one estimation complexity factor that may have biased the original, straight-forward, estimation error analysis: Experience from similar projects. Experience from similar projects is, not surprisingly, strongly connected with decrease of estimation error compared to the situation where the estimator has no previous experience from similar projects (MRE of 0.09 vs 0.20). Interestingly, the estimation model was never used when the estimator lacked relevant experience.

We compared projects where the estimator had experience from similar projects and found that the mean MRE and RE-values were similar for the model and the expert judgment-based estimates (mean MRE of 0.07 vs 0.10 and mean RE of 0.00 vs 0.01) for those projects. This suggests that both estimation methods did fairly well under these conditions, and that the reason for the improved performance of the estimation model was that it was not used in situations with high estimation complexity! The “convincing” story from the straight-forward analysis is therefore not at all convincing when applying a proper error analysis process. A proper conclusion from the estimation error analysis, in our opinion, is that there is no evidence that support a difference in performance between the two estimation methods.

An additional benefit of our analysis is that it provides information about the infrequent use of the estimation model certain situations, e.g., when no experience from similar projects is available. This information suggests that even if the model-based estimates were more accurate, they may not be able to replace the expert judgment-based estimates in all situations.

3. Cost Estimation Studies

In a previous review of estimation terminology [10], we showed that consistent and well-defined terminology is rarely applied in software cost estimation studies. As discussed in Section 2, inconsistent use of terminology is one factor that can lead to flawed conclusions in estimation error analysis unless the studied factor is isolated. In this section we investigate software cost estimation research papers to determine whether isolation strategies are commonly used in analysis of estimation error, i.e. the main purpose of the investigation is to determine whether any isolation attempts are made in the papers, and not to determine the quality of these attempts. In this review, we evaluate our own work and may easily be biased. As we know own work best, these papers might have been selected and evaluated differently than the other papers. Ideally, this part of the review should have been performed by researchers independent of our research group. This was not done for practical purposes. We recommend the readers to examine the papers and make their own opinions.

Selection of reviewed material:

Cost estimation studies were selected from the BESTweb library[*], which is an online library of estimation papers. Papers were selected by reading all the abstracts in BESTweb, and then the full versions of the papers that appeared to meet the following seven inclusion criteria. The paper:

- deals with estimates of software development effort, schedule, budget or cost or with project success/failure/performance,
- analyzes empirical collected estimates from real projects,
- analyzes estimates made before completion of the project,
- analyzes estimation accuracy,
- analyzes cross-company estimation performance,
- is the most recent paper by the main author that met the above criteria, and,
- was reviewed in the terminology review paper [10].

Eight studies met these criteria (see Table 3). The inclusion criteria are based on a combination of practical concerns and what we believe are characteristics of situations where estimation error analyses are most complex.

The small selection of reviewed papers is a threat to the generality of the results. It is outside the scope of this paper to perform a comprehensive review of the estimation literature. Our goal is to give an indication of the state-of-practice for isolation strategies applied in estimation error analysis. We believe the selection is sufficient for this purpose. We are mainly reviewing studies published in journals. It is, for

[*] available at http://www.simula.no/BESTweb

that reason, possible that our selection is biased in the direction of describing a too positive state-of-the-practice in estimation error analysis.

Assessment and data extraction:

We extracted the following information from the studies

- I1: Factor(s) investigated by estimation error analysis.
- I2: Isolation strategies used to isolate the effect of the investigated factor(s).

Possible estimation error analysis factors (I1) were 'estimation ability' (such as the accuracy of an estimation model), 'estimation complexity' (such as ability to control cost) and 'measurement process' (such as estimation error resulting from difference between planned and actual output). Possible isolation strategies (I2) were 'randomization', 'exclusion', 'grouping', 'adjustments' and 'none'. These factors are described in more detail in Section 4. Multiple factors are possible for each study.

For each study the information was extracted by both authors independently of each other. Disagreements were discussed until we reached agreement.

Results:

Table 3 shows the results of the review. Five out of eight studies used no isolation strategy when analyzing estimation error. Among the studies that did use an isolation strategy, grouping is most popular (three studies) followed by exclusion (two studies) and randomization (one study). None of the studies investigated “measurement process”, and none of the studies used “adjustments” as an isolation strategy. All the studies that attempt to isolate the investigated factors investigate factors of type “estimation ability”.

The results indicate that studies often ignore the potential impact of non-studied factors, e.g., how systematic differences in estimation complexity or differences in the measurement process can disturb an analysis of the estimation ability. Many factors may impact the estimation error, and systematic biases are common. Expert estimation may, for example, frequently be preferred to formal models when data from similar projects is missing. Therefore, as demonstrated in Section 2, the lack of proper isolation strategies in the estimation error analysis may be a threat to the validity of the results.

Table 3 Results

Study	**Factor(s) investigated**	**Isolation strategies**
Barki et. al. [2]	Estimation complexity	None
Bergeron and St-Arnaud [3]	Estimation ability, Estimation complexity	Exclusion, grouping,
Heemstra and Kusters [11]	Estimation ability	Grouping.
Jørgensen [14]	Estimation ability	Randomization, exclusion, grouping
Lederer and Prasad [17]	Estimation complexity	None
Moløkken and Jørgensen [19]	Estimation ability, Estimation complexity	None
Ropponen and Lyytinen [21]	Estimation complexity	None
Subramanian and Breslawski [23]	Estimation ability	None

This review does not investigate to which degree it was required to use an isolation strategy to analyze the estimation error factors. It may, for example, be the case that for some analyses, the impact of non-studied factors is ignorable. We believe, however, that in most cases there is a need for isolation of the investigated factor when performing estimation error analysis. An important reason for this belief is that we do not have a deep understanding of when factors do and do not impact the estimation error.

There are large variations in the ways the studies apply isolation strategies. While Bergeron and St-Arnaud [3] excludes projects with less than 150 days (actual effort), Jørgensen [14] limits his analysis to studies of projects with more than 10 hours (estimated effort), and duration of less than approximately eight calendar months. Similarly, Bergeron and St-Arnaud [3] group projects by time of estimation (project phase), Heemstra and Kusters [11] by size, while Jørgensen [14] groups by more variables: Type of task, project complexity, “know-how” skills of the estimators, estimation skills of estimators, whether or not the estimator estimated own work, type of payment, project importance for the customer and organizational role of the estimator.

It is hard to determine whether the studied factor is properly isolated in the three studies that use isolation strategies. It is clear that size of projects and time of estimation impact estimation error, but so does many other factors not addressed in these studies. We believe that this review suggest that there is a potential for improved estimation error analysis also in studies that do use isolation strategies in the estimation error analysis.

4. A Framework For Analysis of Estimation Error

Lack of proper estimation error analysis can have severe consequences as illustrated by the example in Section 2. The review in Section 3 indicates that we frequently measure cost estimation error with improper means to understand what we measure. Particularly, there is a need for clarification of the goal of the estimation error analysis and isolation of the impact of the factors of interest on estimation error. We therefore propose a framework for analysis of estimation error that addresses these issues. The proposed estimation error analysis framework has the following steps:

1. Decide on the factor to be analyzed
2. Define terminology and measures
3. Decide the strategy for isolation
4. Measure estimation error and collect other information necessary for isolation of the factor to be analyzed
5. Analyze and interpret the measured estimation error

Several of the steps are similar to steps included in other measurement frameworks, e.g., the GQM (Goal-Question-Metrics) framework, and textbooks on the improvement of software development processes. Our contribution to the improved analysis of estimation error is the tailoring of the general steps in other measurement frameworks to error analysis in the context of software estimation. A search of the literature has not uncovered a framework for the analysis of software estimation error measurement or an applicable framework for the analysis of estimation error from another domain. We found examples of particular causal models of estimation error, e.g., [1, 18], but these were not, in our opinion, practical frameworks for error analysis in the context of the software industry. We believe that our framework will be particularly useful when the data set is small, e.g., less than 30 observations, and study design involving random allocation to treatment is not possible. These situations, we believe, are typical for analyses of software estimation error.

The proposed framework for the analysis of estimation error requires the measurement of a set of projects and is not meant for the analysis of reasons for estimation error of a single project. For that type of analyses, other types of analysis framework should be applied, e.g., root-cause analysis , post-mortem review [4], or “measured mile” analysis . Even in those cases, however, several parts of our framework may be useful, e.g., as checklists for possible estimation error factors.

The following description of our framework for the analysis of estimation error is not extensive for all steps, e.g., we have not described fully how to ensure proper data collection and statistical analysis. These topics are well covered in standard textbooks on the improvement of software development processes and, therefore, are not covered here. A good estimation error analysis is supported by our framework, but the framework does not replace good measurement and analysis skill and experience.

4.1 Decide on Factor to be Analyzed

The first step in every estimation error measurement and process of analysis should be to get a precise understanding of the factor we want to know more about, i.e., the goal of the study. Ironically, an important step towards a better analysis of the factor that we want to study is to identify those factors that have an effect on the estimation error that we do not want to study. In particular, it is important to identify those factors that we do not want to study (often referred to as “nuisance” factors) that may bias the estimation error analysis. To support this process of factor identification we provide a comprehensive list of factors that can have a major impact on the measured estimation error. The top level categories of factors are these:

1) Estimation ability factors
2) Estimation complexity factors
3) Measurement process factors

Estimation Ability Factors

Important estimation ability factors include the following:

- Expert judgment skills. The ability to estimate the development effort of a software project applying judgment-based estimation methods.
- Accuracy of an estimation model. The ability of a formal estimation model to estimate the development effort accurately. Kitchenham and Linkman summarize different types of sources for estimation model error in [16]: measurement error, model error, assumption error, and, scope error.
- Skills in selection of estimation method. The ability to select the most suitable estimation method.
- Skills in the use of a formal estimation model. The ability to apply formal effort estimation models properly. It may sometimes be useful to separate this ability from the accuracy of an estimation model if the estimation method cannot be assumed to be applied as prescribed.

Estimation Complexity Factors

Some projects are more difficult to estimate than others and the reason for higher estimation error may be a higher estimation complexity. Factors related to estimation complexity include:

- Project management (cost control) ability. The project manager's ability to control the cost, i.e., to manage the project to the budget, is frequently a prerequisite for accurate effort estimates.
- Project member skill. It is usually easier to estimate the effort of skilled developers.
- Client and sub-contractor performance. The performance of a software project depends on the skills of the clients and sub-contractors.
- Completeness and certainty of information. If the input to an estimation process or formal estimation model (measurement error of input variable) is poor, we cannot expect accurate effort estimates.
- Inherent project execution complexity. Innovative projects, e.g., utilizing “leading edge” technology, and projects developing complex functionality, are inherently more difficult to estimate than repeating or simple projects. Another example of inherent project complexity is size (large projects are more difficult to estimate).
- Project priorities. Projects with a strong focus on time-to-market, for example, typically have less accurate estimates than those with a focus on cost control.
- Flexibility in product and process execution. If the project has a flexible scope, a simplification of the product can compensate for initially poor estimates and thus reduce estimation complexity and risk.

The above factors is related to the so-called “cost factors” listed in, e.g., [6, 24]. The main difference between cost and estimation complexity factors is the difference between factors with an impact on cost and with an impact on how difficult it is to estimate. In spite of this difference, it may be useful to examine lists of cost factors to identify sub-factors to the categories above. Factors with an impact on cost, frequently also have an impact on estimation complexity.

Measurement Process Factors

This category covers factors that affect the measured estimation error related to the quality of the measurement process itself:

- Inconsistent use of terminology: When there is a lack of clear definitions of terms and there exist differences in interpretations of important estimation terminology, variance in estimation error cannot automatically be attributed to variance in estimation ability or estimation complexity.
- Logging problems: Lack of proper logging routines for the actual use of effort may result in there being differences in activities included in the measured actual effort, or may affect whether overtime is recorded or not.
- Difference between planned and actual output/process: Software projects may experience increases or reductions in functionality. Similarly, the project may not conduct all planned quality assurance activities or deliver the planned quality. Differences in estimation error may be caused by these differences between planned and actual output/process and not, for example, estimation ability.
- Measurement selection bias: Three particularly important selection biases are: 1) Exclusion of cancelled projects. This typically leads to too positive a view of the estimation ability. 2) Exclusion of projects that have been estimated, but never been started. This may easily lead to too negative a view of the estimation ability, e.g., it is more likely to win a bidding round with an over-optimistic estimate compared with a realistic estimate. 3) Inclusion of only the projects that confirm the desired output of the analysis, i.e., a “confirmation bias”. We discuss the effect of measurement selection biases in [15].

Our categorization of estimation error factors should serve as a starting point for software organizations' discussions on which factors to study, and as a checklist for the need to understand, and control the effect of, the nuisance factors.

4.2 Define Terminology and Measures

Most estimation error surveys and software engineering textbooks do not provide a clear definition of what they mean by an effort estimate. In addition, we, in our role as estimation advisors, have observed an unfortunate mix of interpreting an effort estimate as “most likely effort”, “planned effort”, “best case effort” and “effort used as input to price-to-win” among software professionals (see [13] for examples of why this can hide huge differences in estimation performance). If the data collection does not ensure consistent use of 'effort estimate", it is extremely difficult to perform meaningful analyses.

Two common cost estimation error measures are:

Magnitude of Relative Error (MRE) = |Actual Effort – Estimated Effort| / Actual Effort

Relative Error (RE) = (Actual Effort – Estimated Effort) / Actual Effort

Both measures have been criticized and are far from perfect measures of cost estimation errors. Frequently, it helps to use more than one measure, more than one 'central value', e.g., display both the mean and the median estimation error, and include plots of all the estimation error data.

4.3 Decide the Strategy for Isolation

The following four strategies are candidates for isolation of the estimation error factor of interest:

- Randomization of treatment is the most powerful factor isolation strategy. For example, if we want to compare the estimation performance of two different estimation methods, this strategy would require that the choice of an estimation method be random. The strength of this isolation strategy is that it removes any biased contribution of the non-studied factors to estimation error. One problem with the strategy is that software companies may, for good reasons, not accept the random choice of an estimation method.
- Grouping (Blocking) of the projects into subsets similar with respect to non-studied factors may be necessary to ensure meaningful comparisons. For example, supposing that we want to study the impact of project cost control on the estimation error, we should compare projects with similar levels of estimation expertise and estimation complexity.
- Adjustment for the contribution of one or more non-studied factors to the estimation error may be necessary, if there is no randomization or if the effect of the non-studied factors is not removed through grouping. For example, if some projects delivered more functionality than initially planned, the actual effort values may be adjusted to reflect only the effort to implement the initially planned functionality. The adjustment strategy is only possible if we, to some extent, are able to assess the impact of the non-studied factors on the estimation error.
- Exclusion of observations from the analysis may be necessary when it is possible for the non-studied factors to have a large but unknown impact on the estimation error, or when an observation is not relevant for the goal of the estimation error analysis. Notice that exclusion of an observation does not mean that the observation is of no value for other analyses or purposes. The excluded observation is merely not relevant for the analysis of the factor to be studied.

Most of the time, a combination of isolation strategies, e.g., of grouping, adjustment and exclusion, is needed. If isolation of the factor of interest is not possible, the measurement and the measurement analysis may not be worthwhile.

4.4 Measure Estimation Error and Collect Other Information Necessary for Isolation of the Factor to Be Studied

The information to be collected depends on the factor to be studied (Step 1), the terminology for and measures of estimation error (Step 2), and the chosen isolation strategies (Step 3). In particular, the isolation strategies may have a large impact on the information required for proper analysis:

Randomization: Guidelines for randomization are included in standard statistics textbooks. As opposed to the other isolation strategies, it is not necessary that we collect much context information, as long as the power of the study is high and the treatment is randomized.

Grouping: Proper use of this strategy requires the collection of information about non-studied grouping factors with a large, potentially biased, impact on the estimation error analysis.

Adjustments: Potentially, there is a large number of non-studied factors that could have a substantial impact on estimation error. If we were to group for all these factors, the number of observations in each group would be very small. To avoid this, we may combine grouping with an adjustment strategy. An adjustment strategy requires the identification of the important non-grouped factors and knowledge about their impact on the use of effort.

Exclusion: Projects should be removed from the measurement-based analysis if the impact of non-studied, potentially biasing factors cannot be removed through randomization, grouping or adjustments. Excluded projects may still be subject to analysis, but then other types of analysis frameworks are needed, e.g., root cause analysis or post-mortem review frameworks.

4.5 Analyze and Interpret the Measured Estimation Error

The estimation error analysis to be conducted depends on the measurement goals, the factor to be studied and the isolation strategy chosen. It does not lie within the scope of our framework to provide detailed guidelines for this. Several analysis techniques are described in standard textbooks on statistics and software process improvement.

5. Summary

A real-world example where common analysis of estimation error lead to a flawed conclusion, together with a review of published estimation error analyses in research studies, suggest that there is a need for better analyses of software cost estimation error. Particularly, there may be a need for clarification of the goal of the estimation error analysis and how the impact of the factors of interest on estimation error should be isolated. Currently, we frequently measure cost estimation error with improper means to understand what we measure. We present a framework that can be, we believe, a useful tool to improve the estimation error analysis. In particular, we believe, the checklist to identify non-studied factors with a potential biasing impact on the measured estimation error, the emphasis on proper estimation terminology, and the support on isolation strategies are useful. The framework does, however, not replace good analysis skill.

Acknowledgement:

We thank Professor Barbara Kitchenham for critical and useful comments on an early draft of this paper.

References

[1] T. K. Abdel-Hamid and S. E. Madnick, "A model of software project management dynamics," *proc. COMPSAC 82*, pp. 539-554, 1982.

[2] H. Barki, S. Rivard, and J. Talbot, "An integrative contingency model of software project risk management," *Journal of Management Information Systems*, vol. 17, no. 4, pp. 37-69, 2001.

[3] F. Bergeron and J. Y. St-Arnaud, "Estimation of information systems development efforts: a pilot study," *Information and Management*, vol. 22, no. 4, pp. 239-254, 1992.

[4] A. Birk, T. Dingsøyr, and T. Stalhane, "Postmortem: Never leave a project without it," *IEEE Software*, vol. 19, no. 3, pp. 43-45, 2002.

[5] B. W. Boehm, *Software engineering economics*. New Jersey: Prentice-Hall, 1981.

[6] B. W. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering*, vol. 10, no. 1, pp. 4-21, 1984.

[7] L. C. Briand and I. Wieczorek, "Resource estimation in software engineering," in *Encyclopedia of software engineering*, J. J. Marcinak, Ed., 2nd ed. New York: John Wiley & Sons, 2002, pp. 1160-1196.

[8] P. Coombs, *IT Project Estimation - A Practical Guide to the Costing of Software*. Cambridge: Cambridge University Press, 2003.

[9] J. J. Dolado, "On the problem of the software cost function," *Information and Software Technology*, vol. 43, no. 1, pp. 61-72, 2001.

[10] S. Grimstad, M. Jørgensen, and K. Moløkken-Østvold, "Software Estimation Terminology - The Tower of Babel," *Information and Software Technology*, vol. 48, no. 4, pp. 302-310, 2006.

[11] F. J. Heemstra and R. J. Kusters, "Function point analysis: Evaluation of a software cost estimation model," *European Journal of Information Systems*, vol. 1, no. 4, pp. 223-237, 1991.

[12] C. T. Jones, *Estimating software costs*. USA: McGraw-Hill, 1998.

[13] M. Jørgensen, "How much does a vacation cost?," *Software Engineering Notes*, vol. 28, no. 6, p. 30, 2003.

[14] M. Jørgensen, "Realism in assessment of effort estimation uncertainty: It matters how you ask," *IEEE Transactions on Software Engineering*, vol. 30, no. 4, pp. 209-217, 2004.

[15] M. Jørgensen and S. Grimstad, " Over-optimism in Software Development Projects: “The winner’s curse"," *proc. CONIELECOMP*, 2005.

[16] B. Kitchenham and S. Linkman, "Estimates, uncertainty, and risk," *IEEE Software*, vol. 14, no. 3, pp. 69-74, 1997.

[17] A. L. Lederer and J. Prasad, "Causes of inaccurate software development cost estimates," *Journal of Systems and Software*, vol. 31, no. 2, pp. 125-134, 1995.

[18] A. L. Lederer and J. Prasad, "A causal model for software cost estimating error," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 137-148, 1998.

[19] K. Moløkken and M. Jørgensen, "A review of software surveys on software effort estimation," *proc. International Symposium on Empirical Software Engineering*, pp. 223-230, 2003.

[20] L. H. Putnarn, D. T. Putnam, and W. Myers, "Adapting project estimation to advancing technologies," *American Programmer*, vol. 9, no. 6, pp. 23-29, 1996.

[21] J. Ropponen and K. Lyytinen, "Can software risk management improve system development: an exploratory study," *European Journal of Information Systems*, vol. 6, no. 1, pp. 41-50, 1997.

[22] Sommerville, *Software Engineering*, 7. ed: Addison-Wesley, 2004.

[23] G. H. Subramanian and S. Breslawski, "An empirical analysis of software effort estimate alterations," *Journal of Systems and Software*, vol. 31, no. 2, pp. 135-141, 1995.

[24] C. R. Symons, *Software sizing and estimating - Mk II FPA*. Chichester, U.K.: John Wiley&Sons, 1991.