

Selection of Effort Estimation Strategies

Magne Jørgensen

Simula Research Laboratory & University of Oslo

Abstract: *We currently know little about the factors that motivate the selection and change of estimation strategy in judgment-based effort estimation context. A better understanding of these issues may lead to more accurate judgment-based effort estimates and motivates the four experiments reported in this paper. The experiments' two main results are the identification of the importance of "estimation surprises" (large estimation errors) to motivate estimation strategy change and the large individual variation in the initial choice of estimation strategy. The individual variation seems not only to be a result of differences in previous experiences, but also a result of differences in the mental "accessibility" of the strategies. We found, for example, that the use of a strategy was increased when we instructed a developer to use the same type of strategy on unrelated tasks immediately before. The laboratory contexts of the studies means that the results should be interpreted as a first step towards more knowledge about expert estimation strategies and that there is a strong need for more studies, preferably in field situations, before recommending actions on the basis of the findings.*

1. Introduction

Estimates of the most likely use of software development effort are essential input to planning, bidding, investment analysis, and budgeting processes. These estimates may be based on a variety of estimation strategies, e.g., basing the estimate on the use of the effort of the most similar, previously completed project. The performance of an estimation strategy depends on the context, as illustrated by the diverging results with respect to accuracy of formal estimation models that implement different estimation strategies on different data sets (Shepperd and Kadoda 2001). Hence, it is unlikely that we will ever find one estimation strategy that is superior to all others in all estimation contexts. This means that the ability to select an estimation strategy that is appropriate to the context is essential for accurate effort estimation.

There are many research studies comparing formal effort estimation models, e.g., (Briand, El Emam et al. 1999; Shepperd, Cartwright et al. 2000). We have, however, been unable to find any research studies that compare software developers' strategies in judgment-based effort estimation.

Possible reasons for the lack of studies on how software developers select and change estimation strategy include:

- Judgment-based strategies for effort estimation have not been properly categorized. The difference between formal effort estimation models and judgment-based strategies for effort estimation regarding the issue of their categorization is striking. Formal effort estimation models are divided into numerous categories, dependent on their underlying strategies. All types of effort estimation work in which the quantification step is based on human judgment is on the other hand termed “expert estimation”, typically without any further subdivision into types (Niessink and van Vliet 1997; Jørgensen 2004; Jørgensen 2007). Our extensive search for previous work on judgment-based effort estimation strategies (see (Jørgensen 2005)) yielded only one paper (Rush and Roy 2001). This paper describes the expert judgment process mainly as: “... most judgements are based on the results of referring to historical costs data, and then adjusting up or down accordingly in order to predict the cost of a new project.” However, this study, which describes judgment-based effort estimation as based on a strategy similar to the “anchoring-and-adjustment” heuristic described in (Kahneman, Slovic et al. 1982), does not say anything about how historical cost data is used or how the adjustments are made; neither does it reflect on individual differences with respect to the choice of strategy.
- Judgment-based effort estimation strategies are partly based on unconscious mental processes, and, consequently, are difficult to identify, describe, and analyze in full (Berger 1985; Silverman 1985; Hammond 1996; Hogarth 2005).

The experiments described in this paper try to overcome some of the abovementioned difficulties and shed some light on the individual differences selection and change of strategies for judgment-based effort estimation. The remaining part of the paper is organized as follows:

- Section 2 briefly describes a few relevant theories and findings on cognitive processes and then motivates the estimation strategies and strategy change factors examined in the experiments.
- Section 3 describes and discusses essential experimental design decisions and limitations that are common to all four experiments.
- Section 4 states the research questions addressed in the experiments.
- Sections 5-8 describe the designs and results of Experiments 1-4, respectively.
- Section 9 discusses the results.
- Section 10 summarizes.

2. Cognition and Estimation Strategies

2.1 The Dual-Process Theory of Cognition

A core distinction between reasoning strategies that is commonly made is that between intuitive and analytic processes. This distinction is reflected in the so-called “dual-process” theory of cognition (Evans 2003). This theory suggests that peoples’ judgments reflect the operation of two distinct thinking systems, analysis and intuition, and that these two systems may have a different evolutionary history and neurological basis. Characteristics of analysis and intuition-based thinking are, in accordance with the views reported in (Hammond, Hamm et al. 1987), displayed in Table 1.

Table 1: Characteristics of Analysis and Intuition

Analysis	Intuition
<ul style="list-style-type: none">• High insight into judgment process, and, hence publicly retraceable• Low confidence in outcome, high confidence in method• Input information is objectively evaluated• Slow rate of processing• Few errors, but large when they occur• High cognitive consistency	<ul style="list-style-type: none">• Low insight into judgment process, and, hence difficult to retrace and defend• High confidence in outcome, low confidence in method• Input information is perceptually evaluated• Fast rate of processing• Errors normally distributed• Low cognitive consistency

The two thinking systems interact in many ways, but also compete. The dual-process theory is in common use as a model for understanding judgmental strategies, but is still under debate (Osman 2004). As far as we can see, the debate mainly concerns whether we should consider analysis and intuition as performed by two separate systems or by one system with different types of processes. It is not much debate on the idea that essential judgmental processes are unconscious and sometimes compete with more analytical ones.

To illustrate the relevance of the dual-process theory for judgment-based effort estimation strategies, assume that a software developer is asked to estimate the software development effort for a project. The developer may first try to get an understanding of the problem to be solved. During that process, the developer will come to have a “gut feeling” (“expert opinion” or “intuition”) about the amount of work required. This gut feeling is the developer’s initial judgment-based effort estimate. Assume further that the developer also applies an estimation model or historical data to

estimate the effort and that the analytic process yields an effort estimate that is much higher than his initial judgment-based effort estimate. What should the developer do to resolve this conflict? The gut feeling (which he believes in yet which he knows is sometimes unreliable) tells him one thing, while the analytic process (the reliability of which he also believes in) tells him another. The studies reported in (Aranda and Easterbrook 2005; Jørgensen and Gruschke 2005) suggest that software developers sometimes resolve this conflict between the two competing thinking systems through adapting, consciously or unconsciously, the input to the model, so that the outcome of the analytical thinking is in accordance with their intuition.

Another, frequently observed, consequence of the tendency to believe in the reliability of analysis-based reasoning processes but not in their outcomes is that we tend to describe a judgment-based process as analytical, when it in reality is strongly intuition-based. The study reported in (Shynkaruk and Thomson 2006), for example, reports that we tend to use the analytic process to rationalize and defend judgment-based judgment: *“These data support a model in which initial decisions are made quickly, on the basis of heuristic cues [intuition], and analytic processes are used to justify or rationalize the earlier decisions.”* A tendency to estimate effort based on intuition and then use analytical argumentation to rationalize the estimate may be a major reason for the problems that we and other researchers have had in eliciting descriptions of judgment-based estimation processes from developers. For this reason, studies on judgment-based effort estimation typically cannot base their identification of estimation strategies on asking the estimators about what they did. Neither is it possible to observe the strategies followed by the estimators, because the important steps are mental. Given these difficulties, the current study will be based on the elicitation of estimation strategies indirectly, using an analysis of the estimates themselves as a basis.

2.2 Satisficing, not Optimizing

The limited calculation capacity of the human brain limits the selection of strategies that are available for judgment-based effort estimation. For example, it is unlikely that software developers are able to base their judgmental effort estimates on an optimized least square-type regression analysis of historical data, which is the type of data analysis applied in many regression-based formal effort estimation models. The judgmental strategies have to be based on processes that are simple enough to be processed with sufficient accuracy and speed in real-life situations, i.e., they must follow the “satisficing” rather than the “optimizing” paradigm (Simon 1957). The term typically applied to this type of “satisficing” judgment-based reasoning processes is “heuristics” (or “human judgment heuristics”). The term “heuristic” emphasizes that the reasoning process aims at approximation of the optimal solution and calculation efficiency, and does not require that the optimum solution is found.

Stating the matter in a somewhat simplified manner, the studies on human judgment heuristics suggest that human judgment heuristics work satisfactorily in most contexts, but that their use can lead to systematic biases. As an illustration, the heuristic “anchoring and adjustment” (Kahneman, Slovic et al. 1982) predicts that an estimator will start with a convenient reference value (the anchor) and then adjust the estimate up or down until the estimate feels right. This heuristic may work satisfactorily when the reference value is reasonable, e.g., when the anchor is the typical effort of tasks of the same category or the effort of the closest analogy. However, when the anchor is misleading, the “anchoring and adjustment” heuristic may lead to quite inaccurate effort estimates. We have demonstrated previously how much the presence of misleading anchors affects software developers’ effort estimates (Jørgensen 2007; Jørgensen and Grimstad 2008); this heuristic sometimes may be the cause of inaccurate effort estimates, and the anchor effect may go unnoticed by the software developers. This, in turn, suggests that the selection and use of historical data (the anchors) to some extent are based on unconscious processes, i.e., processes outside the software developers’ awareness.

2.3 Representativeness

In this paper, we focus on how to select judgmental effort estimation strategies from the class of representativeness heuristics. This is a rather broad class of heuristics that assume, when applied to the estimation of work effort, that to some extent people base their estimates on historical information that somehow “represents” the project to be estimated. This representative information may, for example, be the effort of the most similar task (the closest analogy) or the average effort of tasks of the same type. Representativeness heuristics are based on the assumption, which is intuitively easy to accept, that similarity with respect to some properties of a type of thing means similarity with respect to other properties of the same type of thing. Tversky & Kahneman (1982), among others, claim that judgment-based estimates are highly sensitive to representativeness. They also claim that a strong reliance on representativeness may work well in most cases, since similarity with respect to one property frequently means similarity with respect to other properties.

We find it likely that software developers’ judgment-based effort estimates are frequently based on similarity between the task to be estimated and previous tasks and so belong to the class of representativeness heuristics. Reasons for believing this include:

- Organizations tend to over-estimate small and under-estimate large software tasks; see, for example (Jørgensen 1995; Gray, MacDonell et al. 1999), where what constitutes a small or large task is relative to the tasks typically completed by a developer or by an organization. This bias is in accordance with what we would expect, given the use of a representativeness-based estimation strategy. When estimating a larger than usual task it is more likely that the

representative tasks, e.g., the most similar or the average effort of tasks of the same type, are smaller than the task to be estimated. This, together with the observed lack of adjustment for differences (see, for example, the anchoring and adjustment-findings in (Kahneman, Slovic et al. 1982)), leads to under-estimation. This consequence of the use of the representativeness heuristic is as predicted in, for example, (Kahneman and Tversky 1973).

- The heuristics have been found to be important for other quantitative estimation tasks. A summary of its relevance for other forecasting domains can be found in (Harvey 2007).
- The representativeness heuristics may be relatively easy to use without tool support in software development effort estimation contexts, because the required input information is frequently available and the heuristics do not require complex, calculation-intensive processes. This fits the “satisficing”, not “optimizing” findings.

We do not believe or assume that strategies belonging to the representativeness heuristics are the only types of strategy available for judgment-based effort estimation in typical software development contexts. It is highly likely that other heuristics and effects can be involved, too, such as “accessibility” (the most recent projects or strategies used are emphasized because they are most vividly remembered) (Mussweiler 2003), “anchoring-and-adjustment” (Northcraft and Neale 1987), and “wishful thinking” (Jørgensen and Sjøberg 2001).

2.4 Categorization of Strategies

Strategies based on the use of the representativeness heuristics differ in what is used as representative information. For the purpose of our experiments, we chose to examine the following strategies:

- **Closest analogy productivity (CIAn):** When applying the closest analogy strategy we assume, for the purpose of the analyses of our study, that only highly similar projects (two or three very similar projects in Experiments 1, 2, and 4, and, the single most similar project in Experiment 3) are selected as the basis for the effort estimation of the new project. The projects vary in size and we therefore assume that it is not the effort, but the productivity, of the closest analogy that is used as the representative value. If the productivity of the closest analogy is, for example, 40 lines of code per work-hour, then the new project will be assumed to have this productivity, as well.
- **Aggregated productivity (AggAll and AggClass):** When applying the class average strategy, we assume that the software developers decide what the appropriate class of relevant projects is and use the average productivity of the projects in that class as the basis for the effort estimate of the new project. In our experiments, we are mainly concerned about the use of estimation strategies based on the decision that the class of relevant projects includes all previously

completed projects (AggAll) or only the class of similar projects (AggClass), e.g., those with the same type of application and/or development environment. Notice that the separation of AggAll and AggClass is tailored to the purposes and data sets of this study. In principle, AggAll belongs to AggClass, but uses a larger class of projects.

On a scale from highly specific representative information to fully aggregated information, the pure use of the CIAn strategy and AggAll are the two extremes. There are contexts in which it is, to a great extent, arbitrary whether we term a strategy analogy- or aggregation-based, e.g., a situation in which the representative productivity is based on three projects of almost the same type. Our classification of strategies is only meant to be a useful means to better understand the selection and change of strategies. It is not meant to provide a general framework. We will try to design the experiments so that we will, in most cases, be able to identify which of the strategies is likely to have dominated the effort estimation process.

The importance of use of closest analogy and category-based estimation strategies are supported by results in the two studies we report in (Jørgensen 2005). The analysis of the usage of estimation strategies that we completed in these two studies was based on video recording of the estimation team's discussions and the developers' own descriptions. Although the main finding of the two studies was that the software developers were, in most cases, unable to describe their (largely unconscious) judgment-based estimation strategy, we nevertheless found some evidence suggesting the use of closest-analogy and aggregated productivity-based rules. Two examples from the estimation team's discussions, which provide evidence in favour of the productivity-based estimation strategies that we describe as CIAn and AggClass in this paper, include:

- *“The use of [name of the development tool] is not very efficient. Even simple reports take time. I remember I estimated two work-days per report [in another project].* Here the developer points to one previous project in which, assuming that his estimates were correct, the productivity was two work-days per report. This seems to be similar to what we describe as CIAn.
- *“I have this simple rule in my head: Simple user screens are 6 work-hours, medium are 15 and difficult are 60, and if it's extreme you have to add much more.”* The mentioned rule seems to be the developer's own aggregated, perhaps mean, productivity of similar tasks of various complexity. This seems to be similar to what we describe as AggClass.

2.5 Estimation Strategy Selection Factors

In our experiments, we focused our analysis on the idea that the following three factors may be important drivers of the selection and change of estimation strategies:

- Strategy preference, e.g., a preference for a particular strategy that is based on a software developer's previous success and failure when following different strategies. Underlying a

strategy preference is also a cost-benefit analysis of following a strategy; see, for example, (Risekamp and Otto 2006) for a discussion of this.

- Strategy failure, e.g., the experience of a large estimation error (“estimation surprise”) when applying a strategy. The importance of this factor for strategy change is discussed in, for example, (Wills, Lavric et al. 2007).
- Aggregated performance in the current context, e.g., the average estimation error of the different strategies.

To a large extent, it is common sense that these three factors should be analyzed. Strategy preference will be especially important for the initial choice of estimation strategy, while strategy failure and aggregated performance will be essential to motivate and guide strategy change. In some sense, we expect the software developers to follow a process similar to a Bayesian inference process (Berger 1985), where both prior belief (preference) and current evidence (performance of a strategy in current context) are weighted.

There may be additional factors that lead to individual differences in strategy preference and selection, e.g., factors that are based on differences in the willingness to use the additional mental effort required to aggregate class values in comparison to the closest analogy strategy (Cacioppi and Petty 1982; Levine, Huneke et al. 2000). These factors will not be emphasized in our analyses, but will be viewed as underlying reasons for the other factors, e.g., the preference factor.

3 General Study Design Decisions and Limitations

3.1 Design Decisions

As noted above, the unconscious nature of essential steps of judgment-based effort estimation means that software developers are typically not very good at explaining their estimation strategies. This is illustrated by our numerous failed attempts to gain access to the estimation strategies through questionnaires, interviews, logging, and think-aloud protocols; see (Jørgensen 2005) for a summary. Given this experience, we decided to design experiments where it would be possible to derive the use of the strategies ClAn, AggClass and AggAll from the estimates alone, rather than having to rely on the participants’ own descriptions of the use of estimation strategies.

There are individual differences in how people solve most tasks that involve reasoning, problem solving, or judgment (Roberts 2000; Roberts and Newton 2001). This means that we cannot assume that there is one understanding or model of behavior that explains all software developers’ selection and change of strategy well. This observation is a major motivation for our experiments’ analyses of individual differences in the choice and change of effort estimation strategies.

The estimation strategies of inexperienced (e.g., computer science students) and experienced software developers are probably not the same. Our results reported in (Jørgensen and Sjøberg 2004), for example, suggest that even when the effort estimates of these different groups of developer are the same, they will typically be based on quite different analyses and assumptions. For this reason, we decided to include as participants in our experiments only professional software developers who had previous experience of software development and estimation.

3.2 Study Limitations

The decision to design experiments such that it would be possible to identify the estimation strategies from the effort estimates alone resulted in our deviating from typical real-life estimation situations in a number of respects. Real-life estimation situations typically include the reading of a requirement specification and then applying one's own experience from similar development projects to estimate the most likely effort. However, the inclusion of a real-life requirement specification would make it very difficult to identify what actual information was used and how it was used. In such a situation, it would be difficult to identify the estimation strategy on the basis of the effort estimate alone. To enable the strategy to be identified, we had to remove most of the project information and make it likely that the software developers used mainly the historical information and, to a lesser degree, their personal experience of software development. For this reason, the information about the projects in our experiments included only information about a small number of factors that drive the cost, e.g., only about the estimated size of the software and the type of development environment.

The reduction of project information to only a few cost drivers makes the estimation situation different to most, but not all, real-life estimation situations. This means, however, that we should be careful when extrapolating the results to real-life estimation contexts, in which much more information about the project to be completed is available (including, for example, a partial technical understanding of how to solve the projects and a greater amount of information about similar projects that had been completed earlier.) On the other hand, the strategies that we studied are implementations of the basic processes of most judgments. Adding more information about the current and the previous projects does not, for example, change the need for a (perhaps unconscious) decision on whether the judgment should rely on specific (e.g., closest analogy) or aggregated (e.g., class average) information about the projects.

In field settings, it is typically the case that a greater amount information is available about the project to be estimated, but that historical data is less readily available. This may lead to less analytical estimation processes and less use of aggregated historical data than in the processes used in our experiments.

Finally, the software developers who participated in the experiment were estimating the effort of projects of another company and not their own work. In a previous study (Jørgensen and Gruschke 2008) we obtained results that suggest that software developers are much more willing to apply historical data presented to them when estimating other people’s work than their own, perhaps as a result of a removal of the “wishful thinking” bias. While the decision to use specific or aggregated information is still believed to be an essential part of the estimation process when estimating one’s own work, it is probable that many other competing processes would be involved, too.

In total, there are essential limitations, motivated by a need for internal validity (identification of estimation strategies) that have a negative effect on the external validity of the results. That being so, the results should be interpreted principally as a first step towards a better understanding of the selection and change of judgment-based estimation strategies, and as requiring replication in and support from field-based studies.

4. Research Questions

The four experiments were designed to answer the following research questions:

- RQ1: When there is no information clearly in favor of an estimation strategy, how are the individual preferences of the software developers distributed with respect to use of the strategies based on closest analogy and aggregation?
- RQ2: What is the relative importance and role of the factors driving strategy selection and change that were described in Section 2.5? (i.e., the factors strategy preference, estimation surprise, and average accuracy in the current estimation context.)
- RQ3: Is the initial choice of estimation strategy (the estimation strategy preference) easily biased by irrelevant factors?

Experiments 1, 2, and 3 address the RQs 1 and 2, while Experiment 4 addresses RQ3.

5. Experiment 1

5.1 Estimation Context

As discussed in Section 3, we had to reduce the information about each project and to enable the estimation strategies to be estimated from the estimates. The estimation context we used as starting point for Experiment 1 was the following:

”Assume that you are the project leader in a software development organization and are asked to give a rough estimate of the productivity of a new project (P-NEW). You search in the

organization's own project database of previously completed projects and find two projects with the same development platform, type of application, size, and complexity (the project P-35 and P-48) as P-NEW; see Table 2 below.

Table 2: Characteristics of Most Similar Projects

Project	Development platform	Type of application	Size	Complexity	Productivity
P-35	Java	Client/server (Sybase)	40 use cases	MEDIUM	20 work-hours/use case
P-48	Java	Client/server (Sybase)	30 use cases	MEDIUM	40 work-hours/use case
P-NEW	Java	Client/server (Sybase)	38 use cases	MEDIUM	? work-hours/use case

The project database (where you found P-35 and P-48) consists of 50 previously completed software development projects in which the number of use cases is a relevant indicator of the size of the delivered software. From this database, the following information is derived:

- The projects vary in productivity from 10 to 200 work-hours per use case.
- The mean productivity, based on all projects in the project data base, is approximately 80 work-hours per use case.
- The mean productivity of the 10 projects that have the same development platform, type of application, and complexity (the three assumed most important productivity factors) as P-NEW is approximately 70 work-hours per use case. P-35 and P-48 are included in those 10 projects.

As can be seen from Table 2, a high productivity estimate (around 30 work-hours per use case) would suggest that the software developer based it predominantly on the CIAn (the very similar projects P-35 and/or P-48), a medium-high productivity estimate (around 70 work-hours per use case) that the dominant strategy was AggClass, and a low productivity estimate (around 80 work-hours per use case) that it was based on AggAll. Values between the productivity predicted by the use of a strategy would suggest a combination of strategies. In our experiment, we were mainly interested in the strategy that dominated the estimation process and, for this purpose, introduced the following strategy identification rule, where the boundary values (50 and 75) are the mid-values between the values predicted by use of a strategy:

```
IF ESTIMATE < 50 THEN STRATEGY = "CIAn"  
ELSE IF ESTIMATE < 75 THEN STRATEGY = "AggClass"  
ELSE STRATEGY = "AggAll"
```

Opinions may differ about what the rational choice of strategy in the described situation should be. For example, one could argue that the productivity of the two closest analogies (P-35 and P-48) is unusually high and should therefore not be emphasized. However, it is also possible to argue that these two projects seem to be very similar to P-NEW and should therefore be emphasized. In our opinion, there is no obvious normative behaviour in this situation. We therefore assume in our analyses that the developers' choices reflect, to a large extent, their personal, perhaps unconscious, preferences in similar situations, e.g., that the strategy choice is based on previous experience from applying estimation strategies in similar situations.

5.2 Study Design

The participants in Experiment 1 were 74 software developers from various companies who were attending a software estimation seminar. The participants:

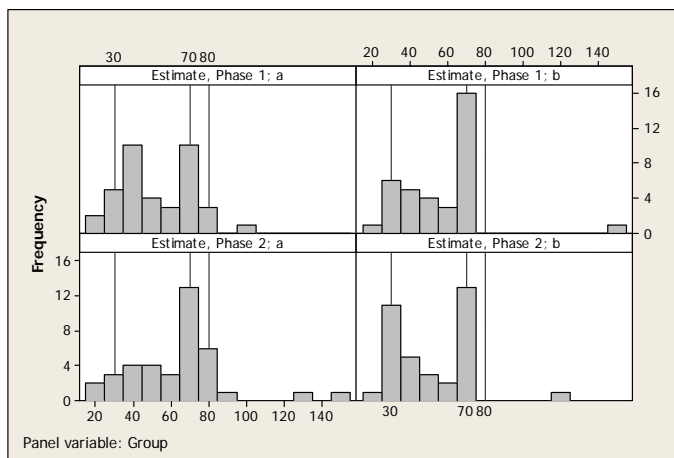
1. were divided randomly into two groups (Group A and Group B),
2. read the description of the estimation context
3. answered the question: "*What would be your estimate of P-NEW's productivity?*" (Phase 1 estimate)
4. received additional information:
 - a. Group A: "*Assume that you receive the information that estimates based on the productivity of the 2-3 most similar projects have been, on average, very inaccurate. Update your estimate (if necessary), using this information as a basis.*"
 - b. Group B: "*Assume that you receive the information that estimates based on the productivity of the 2-3 most similar projects have been, on average, very accurate. Update your estimate (if necessary), using this information as a basis.*"
5. were requested to update the previous estimate (Phase 2 estimate)

The different information that was given to Groups A and B enabled us to examine the strength of the impact of the aggregated information about an estimation strategy in comparison to the initial preference on the choice of estimation strategy. If the aggregated information about the CIAn strategy, i.e., that the 2-3 most similar projects had, on average, been either very good or bad strategy, had a large impact, the developers in Group A would provide estimates close to AggClass or AggAll and the developers in Group B would provide estimates close to CIAn in their updated estimates (Phase 2 estimates).

5.3 Results

The histograms of the resulting productivity estimates are displayed in Figure 1. The vertical lines are the midpoints of the two closest analogies (30 work-hours per use case), the mean of the projects of similar type (70 work-hours per use case), and the mean of all projects (80 work-hours per use case), i.e., the values corresponding to CIAn, AggClass and AggAll, respectively. The upper left histogram depicts the estimates of Group A in Phase 1, the upper right the estimates of Group B in Phase 1, the lower left the estimates of Group A in Phase 2, and the lower right the estimates of Group B in Phase 2.

Figure 1: Histogram of Productivity Estimates for Experiment 1



The developers' assumed strategy selections and changes, as derived from the rule for identifying strategies, are displayed in Tables 3 (Group A) and 4 (Group B).

Table 3: Strategy Selection and Change Group A

From strategy (row) To strategy (column)	CIAn	AggClass	AggAll	Sum: Phase 1
CIAn	10	5	3	18
AggClass	0	14	2	16
AggAll	0	0	4	4
Sum Phase 2	10	19	9	38

Table 4: Strategy Selection and Change Group B

From strategy (row) To strategy (column)	CIAn	AggClass	AggAll	Sum: Phase 1
CIAn	12	0	0	12
AggClass	6	17	0	23
AggAll	0	0	1	1
Sum Phase 2	18	17	1	36

An examination of the estimates provided in Phase 1 (see Figure 1 and Tables 3 and 4) suggest that there were substantial differences in strategy preference, given the same estimation context. Thirty developers (18 in Group A and 12 in Group B) chose estimates that, according to our strategy identification rule, were assumed to be dominated by the CIAn strategy, 39 (16 in Group A and 23 in Group B) assumed to be dominated by the AggClass strategy, and, 5 (4 in Group A and 1 in Group B) assumed to be dominated by the AggAll strategy in Phase 1. In other words, 40% of the software developers seem to have selected a strategy that is dominated by the CIAn strategy, 53% a strategy that is dominated by the AggClass strategy, and 7% a strategy that is dominated by the AggAll strategy, in an estimation situation in which they received exactly the same information. This is, of course, only a weak indication of what would happen in more realistic estimation situations, but suggests nevertheless that there are, in some contexts, major individual differences in preferences for estimation strategy. Interestingly, our categorization of estimation strategies based on the estimation team’s discussions and the developers’ own descriptions (Jørgensen 2005), indicates a similar distribution of preferences for the use of an aggregation-based and analogy-based, estimation strategy.

An examination of the changes in strategy from Phase 1 to Phase 2 reveals that eight out of the 18 (44%) developers in Group A who chose CIAn in Phase 1 changed to a more aggregation-based strategy (AggClass or AggAll) in Phase 2, as a response to the information that using the CIAn led, on average, to very inaccurate effort estimates. Six out of the 24 (25%) developers in Group B who chose AggClass or AggAll in Phase 1 changed to the CIAn strategy in Phase 2, as a response to the information that using the CIAn led, on average, to very accurate effort estimates. These results indicate that aggregated historical performance may not drive strategy change very strongly. It is also interesting to note that a change from CIAn to AggClass seem to be easier than a change from AggClass to CIAn. A possible cause may be that the individual preference related to the AggClass strategy is stronger than that related to the CIAn strategy.

6 Experiment 2

6.1 Design

The estimation context (Phase 1) of Experiment 2 was designed to be exactly the same as that of Experiment 1; see Section 5.1 for a description. The purpose of Experiment 2 was to replicate and extend the results in Experiment 1:

1. It examined the robustness of the estimation strategy distribution with the same estimation task, but a different sample of software developers
2. It examined whether an “estimation surprise” (i.e., the information that AggClass and AggAll turn out to be inaccurate, and ClAn accurate on the project P-NEW) results in an increase in the proportion of software developers who change their strategy from AggClass to ClAn in comparison to the proportion of developers doing this in Experiment 1. Rationally speaking, it is difficult to defend the position that one project observation (one estimation surprise) should be given more weight than the aggregated accuracy of a strategy. However, results reported in, for example (Wills, Lavric et al. 2007) suggest that estimation surprises may be more important for strategy change than aggregated historical data.

The participants in Experiment 2 were 55 software developers who were attending a seminar on software development effort estimation. There was no overlap in the populations of Experiments 1 and 2. The participants:

1. read the description of the estimation context (which was identical to that in Section 5.1)
2. answered the question: “*What would be your estimate of P-NEW’s productivity?*” (Phase 1)
3. received the outcome that P-NEW actually had a productivity of 30 work-hours per use case. This meant that the developers who used AggClass and AggAll would experience a large estimation error (estimation surprise), while those who used ClAn would get confirmation about the accuracy of their strategy.
4. were asked to update the estimated productivity of a new project (P-NEW-2) with properties as described in Table 5.

Table 5: Characteristics of Most Similar Projects

Project	Development platform	Type of application	Size	Complexity	Productivity
P-35	Java	Client/server (Sybase)	40 use cases	MEDIUM	20 work-hours/use case
P-48	Java	Client/server (Sybase)	30 use cases	MEDIUM	40 work-hours/use case
P-NEW	Java	Client/server (Sybase)	38 use cases	MEDIUM	30 work-hours/use case
P-NEW-2	Java	Client/server (Sybase)	30 use cases	MEDIUM	? work-hours/use case

6.2 Results

We found that:

- The individual differences in choice of strategy were about the same as in Experiment 1, Phase 1; see the rightmost column in Table 6.
- Most of the software developers who applied AggClass (18 out of 22) or AggAll (5 out of 6) in Phase 1 (on P-NEW-1) changed to CIAn in Phase 2; see Table 6. This is a much larger proportion than the change we observed from AggClass and AggAll to CIAn in Experiment 1. In Experiment 1, only 25% of the developers who received the information that CIAn led to accurate effort estimates changed from AggClass or AggAll to CIAn, while in Experiment 2 as many as 82% changed (23 out of 28). This suggests that experiencing an estimation surprise (a high estimation error) may be a much stronger factor for strategy change than aggregated historical information about the estimation strategy. This finding was examined further in Experiment 3.

Table 6: Strategy Selection and Change

From strategy (row) To strategy (column)	CIAn	AggClass	AggAll	Sum: Phase 1
CIAn	26	1	0	27 (49%)
AggClass	18	4	0	22 (40%)
AggAll	5	1	0	6 (11%)
Sum Phase 2	49 (89%)	6 (11%)	0 (0%)	55

7 Experiment 3

Experiment 3 was designed to further examine the relative importance of the three factors driving a change in estimation strategy: strategy preference, documented performance in the current context, and estimation surprises. Experiments 1 and 2 suggested that the individual estimation strategy preferences vary a lot and that estimation surprises may be a much stronger factor driving strategy change than aggregated performance in the current estimation context. Experiment 3 tried to replicate and extend those results with a new data set, a new estimation context, and a new set of developers.

7.1 Design

Twenty-nine software developers and project managers of a Norwegian e-commerce software development organization were paid close to their ordinary fees per hour to participate in Experiment 3. There was no overlap in population with the participants of Experiments 1 and 2.

We identified 15 real-world projects of a Norwegian software development company that were useful for the purpose of our study. To allow us to identify the estimation strategy on the basis of the estimate alone, we limited the information about each project to include only information about the estimated size (in lines of code estimated before the project began) and the work environment (the development language) of the project. The data set is displayed in Table 7.

Table 7: Project Data Set

Project Id	Estimated Size (LOC)	Development Environment	Actual effort (man-months)
1	30000	COBOL	60
2	30000	C/C++	40
3	20000	COBOL	60
4	54000	JAVA	30
5	62000	COBOL	200
6	28000	C/C++	20
7	35000	JAVA	15
8	30000	JAVA	30
9	48000	C/C++	30
10	93000	JAVA	50
11	57000	JAVA	25
12	22000	JAVA	10

13	24000	C/C++	40
14	42000	C/C++	40
15	40000	JAVA	40

The estimation information was provided in a Microsoft Excel spreadsheet. This enabled values such as mean productivity (LOC/man-month) JAVA-projects to be calculated easily by the participating software developers. The following steps were completed by each of the 29 software developers:

1. Opening of the spreadsheet. The spreadsheet included full information (estimated size, development environment, and actual effort) about the five first projects (Project Id 1-5), and the estimated size and development environment of Project 6; see Figure 2.
2. Reception of written instructions about the use of the spreadsheet. This included information about the variables and the sequence of the work, and a guarantee that the data would be treated as confidential information.
3. Estimation of the most likely effort of Project 6. To submit the estimate, the software developer had to insert “y” in the “Estimate submitted?” column. As soon as the estimate was submitted, the actual effort of the project and the estimation error were displayed. At the same time, the information (estimated size and development environment) of the next project, i.e., estimation information about Project 7, appears in the spreadsheet.
4. Step 3 was reiterated for sequential Project IDs until the estimation of Project 15 was completed. The software developers were free to do the necessary calculations in the spreadsheet, on a piece of paper, or elsewhere.

We logged the work sequence of the software developers to see whether they worked as instructed and did not, for example, change their estimate after they knew the actual effort. The log showed no deviation from the prescribed behavior.

Figure 2: The Estimation Worksheet Before the First Estimate

Project Id	Estimated Size (LOC)	Development Environment	Your Estimate of Most Likely Effort	Estimate submitted?	Actual effort (man-months)	Estimation Error
1	30000	COBOL	X	X	60	X
2	30000	C/C++	X	X	40	X
3	20000	COBOL	X	X	60	X
4	54000	JAVA	X	X	30	X
5	62000	COBOL	X	X	200	X
6	28000	C/C++				

There are many measures of estimation error; see (Jørgensen 2007) for an overview and a discussion. In Experiment 3 we applied the measure BRE (Balanced Relative Error), see (Miyazaki, Terakado et al. 1994), instead of the more common MRE (Magnitude of Relative Error), because BRE provides more symmetric values for estimation error, such that, for example, an estimation overrun is weighted just as much as an estimation underrun. MRE penalizes effort underruns more than effort overruns, which we think is not in accordance with how software developers would interpret effort estimation errors. BRE may therefore be more suitable as an indicator of when, from the developers' perspective, an estimation strategy should be replaced with another that is based on aggregated estimation accuracy. However, which measure of estimation accuracy is chosen is not critical and we observed that an analysis based on the use of MRE would give similar results.

$$\text{BRE} = |\text{Actual effort} - \text{Estimated effort}| / \min(\text{Actual effort}, \text{Estimated effort})$$

$$\text{MRE} = |\text{Actual effort} - \text{Estimated effort}| / \text{Actual effort}$$

7.2 Estimation Strategy Analysis

7.2.1 Data Set Properties

The following properties of the data set in Table 7 are, among others, relevant for the choice and change of estimation strategy:

- **AggAll is not an accurate strategy:** There is a clear difference in the productivity as measured by estimated LOC per work-hour for projects completed in COBOL, C/C++, and JAVA and these differences are identifiable early in the estimation work. As an illustration,

summarizing the productivity of the first 10 projects gives a mean productivity of 381 LOC/work-hour for COBOL projects, 1250 LOC/work-hour for C/C++ projects, and 1748 LOC/work-hour for JAVA projects. This means that the software developers could easily determine that the strategy AggAll would not work well in comparison to AggClass and CIAn.

- **There is no, or little, necessity to make adjustments with respect to productivity according to project size or learning:** Two possible adjustments with respect to productivity are based on (i) the size of the project and (ii) learning over time. We analyzed the effect of these two factors and found only a weak positive correlation between size and productivity (0.3) and no improvement over time.
- **CIAn is the natural choice for the first estimates:** For the two first estimates, i.e., the effort estimates of Projects 6 and 7, there was only one previous project that was conducted using the same type of development environment. The CIAn for these projects is therefore a natural choice of strategy, and nearly all the software developers used this strategy on these tasks. Use of the AggAll strategy is also possible, but as discussed earlier, there were already indications of substantial differences in productivity between the different development environments.
- **There are “estimation surprises”:** The historical data and feedback of the first estimates show that using the closest analogy strategy (CIAn) leads to the effort for some projects being estimated quite inaccurately; hence, there will be prediction surprises that motivate a change of strategy from CIAn. In particular, it will be rational to assess whether the category average strategy (AggClass) should be used instead of CIAn. Use of the other strategies will also lead to estimation surprises.
- **The best strategy on the last two tasks, according to the aggregated estimation accuracy, is AggClass:** As more information about the estimation accuracy becomes available for all estimation strategies, it is possible to compare the aggregated accuracy of the estimation strategies. While the use of CIAn is a defensible choice on most projects, based on aggregated estimation accuracy, the AggClass strategies has the best aggregated estimation accuracy (best median BRE) when estimating Projects 14 and 15. If changes to the estimation strategy used were based purely on aggregated accuracy, most changes from CIAn to AggClass should appear here. Whether this happened or not will be discussed in Section 7.2.3.

7.2.2 Identification of Strategies

To simplify our strategy analysis, we transform the estimated effort to the corresponding estimated productivity. If, for example, a developer estimates the effort to be 28 man-months and the estimated size is 28000 LOC, the estimated productivity is $28000/28$ LOC/man-month = 1000 LOC/man-month.

Table 8 provides information about the productivity estimates that we would assume if a developer used one of the strategies. The rule that we used to identify the dominating strategy is that the dominating strategy is the one with the smallest difference between the productivity derived from the use of the strategy and the developer's estimated productivity. If, for example, the estimate is closest to that derived from the use of the CIAn strategy, we assume that the CIAn strategy is the dominating strategy.

Table 8 Estimated Productivity (LOC/man-month) per Strategy

Project Id	CIAn	AggClass	AggAll
6	750	<not relevant>	739
7	1800	<not relevant>	849
8	2333	2067	1061
9	1400	1075	1053
10	1800	1711	1114
11	1800	1748	1189
12	1000	1855	1288
13	1400	1250	1364
14	1600	1088	1305
15	2333	1912	1287

7.2.3 Estimation Accuracy per Strategy

Tables 9 and 10 describe the BRE values connected with the use of the different strategies. Table 9 shows where the estimation surprises will be a particular strategy is followed. Table 10 shows the strategy that would be chosen if the aggregated accuracy, measured as the median BRE, on comparable projects would dominate the selection and change of estimation strategy. We use these data in our analysis of the selection and change of estimation strategies. The median BRE is used instead of the mean BRE because the median is more robust towards outliers and because a median would be simpler to calculate by the software developers and hence may be more likely to be used as an aggregation measure. However, the results would be little affected by a change from the median BRE to the mean BRE or MRE-based measures.

Table 9 Estimation Error (BRE) per Strategy and Project

Project Id	CIAn	AggClass	AggAll
1	<not relevant>	<not relevant>	<not relevant>
2	<not relevant>	<not relevant>	0,50
3	0,50	<not relevant>	0,88
4	<not relevant>	<not relevant>	2,41
5	0,61	<not relevant>	1,73
6	0,87	<not relevant>	0,90
7	0,30	<not relevant>	1,75
8	1,33	1,07	0,06
9	0,14	0,49	0,52
10	0,03	0,09	0,67
11	0,27	0,30	0,92
12	1,20	0,19	0,71
13	1,33	1,08	1,27
14	0,52	0,04	0,24
15	1,33	0,91	0,29

Table 10 Median BRE of the Estimation Strategies

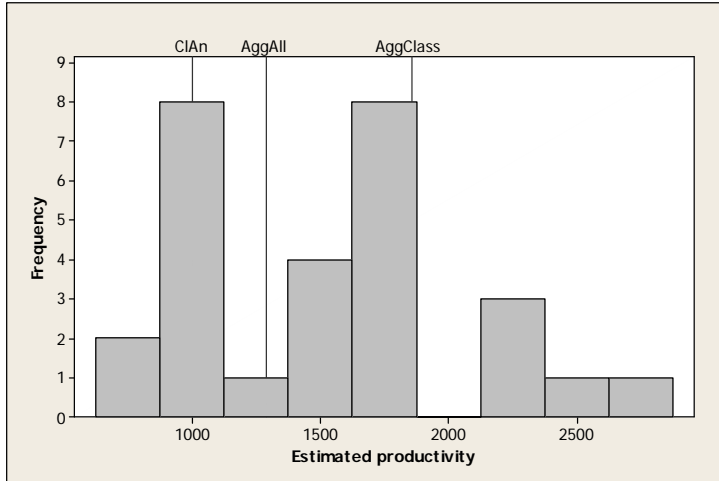
When estimating	Based on median BRE of	Median BRE of CIAn	Median BRE of AggClass	Median BRE of AggAll
Project 9	Project 8	1,33	1,07	0,06
Project 10	Projects 8-9	0,74	0,78	0,29
Project 11	Projects 8-10	0,14	0,49	0,52
Project 12	Projects 8-11	0,20	0,40	0,59
Project 13	Projects 8-12	0,27	0,30	0,67
Project 14	Projects 8-13	0,73	0,40	0,69
Project 15	Projects 8-14	0,52	0,30	0,67

7.2.4 Limitations of the Analysis

We found that it was not always easy to be confident about the actual use of an estimation strategy. This is illustrated in Figure 3, which shows a histogram of the estimated productivity of Project 12. It may be seen that there are many estimates around CIAn and AggClass, which indicates that those were the two dominant estimation strategies for that project. However, several of the estimates are difficult to analyze. For example, does estimating an effort value that corresponds to a productivity of 1300 mean that the software developer has combined CI1 and AggClass, or that the developer used the AggAll strategy? Furthermore, none of our estimation

strategies would lead to estimates higher than 2000. How can we explain these values? Are they based on AggClass and an assumption of learning, calculation errors, or something else? These limitations mean that our analysis should rely on patterns that seem to be stable over several projects and not so much on the analysis of single projects.

Figure 3: Histogram of Estimated Productivity of Project 12



7.3 Results

Table 11 provides an indication of the strategies selected for each project.

Table 11: Strategy Selection

Project	ClAn	AggClass	AggAll
6	25	0	3
7	24	0	4
8	11	12	5
9	9	13	6
10	0*	19	9
11	9	11	8
12	10	14	4
13	5	18	5
14	1	25	2
15	3	23	2

*) As can be seen in Table 6, the use of ClAn and AggClass lead to almost the same estimated productivity. As far as we know, there is no good reason for a change from ClAn to AggClass for this project and back to ClAn for the following project. Consequently, it

is likely that the proportion of strategy use is similar to the tasks immediately before and after, i.e., that the distribution for Project 10 is misleading.

Table 11 shows that nearly half of the software developers change to values corresponding to dominance of the AggClass strategy as soon as it is possible to use the AggClass strategy, and before there is enough data to evaluate the strategy, i.e., data for Project 8. This observation supports the findings in Experiments 1 and 2 to the effect that aggregation-based strategies are preferred by many software developers.

The role of estimation surprises seems to be important in the developers' change of strategy. That being so, the results of Experiment 3 support the findings of Experiments 1 and 2. For example, the second wave of changes from CIAn to AggClass follows the large estimation errors when applying CIAn to Projects 12 and 13, i.e., when estimating Projects 13 and 14. Another indication of the role of estimation surprises is that seven out of the 12 developers who changed from CIAn to AggClass on Project 8 changed back to CIAn on Project 9, due to the high estimation error of Project 8 when using AggClass. Knowing that CIAn would have led to an even larger estimation surprise on that project, this supports our assumption that estimation surprises drive software developers to change their strategy. A third indication of the importance of estimation surprises is the observation that the software developers experienced, in total, 40 estimation surprises (here defined as $BRE > 1,0$). As many as 31 of these surprises led to a change of strategy. If we change the level of BRE defined as estimation surprises to include all estimates with more than a 50% error (i.e., $BRE > 0,5$), the pattern is similar, although less extreme. In this case, 75 out of the 119 estimation surprises led to a change of strategy.

There are reasons to believe that in many cases, estimation surprises are the direct cause, and not only a correlating factor, of changes of estimation strategy. If the aggregated estimation accuracy of a strategy was the main guiding principle for selecting an estimation strategy, we should expect most software developers to change from CI1 to AggClass when the aggregated accuracy, e.g., median BRE, of AggClass was better, but not before. As reported earlier, see also Table 10, this would not be the case before the estimation of Project 14. Consequently, the actual selection and change of strategy deviate substantially from the historical performance-based pattern. This deviation cannot be described fully by an appeal to personal strategy preference, because strong strategy preferences would lead to fewer, rather than more, strategy changes. A full description may need the additional factor of a strong impact from estimation surprises.

8. Experiment 4

In Experiment 4, we tested the stability of estimation strategy preference. We did this by testing whether the mental “accessibility” of an estimation strategy is an essential factor for preference. An extensive review of studies on mental accessibility can be found in (Mussweiler 2003).

The experimental design was based on increasing the accessibility of a strategy by applying so-called priming. Priming refers to “*activating parts of particular representations or association in memory just before carrying out an action or task*” (www.wikipedia.org). Priming effects are observed in many fields and are well-documented; see for example (Schacter, Dobbins et al. 2004) for a review.

In Experiment 4 we will try to artificially activate (prime) the representation or association in memory related to use of analogy and aggregation-based strategies, and see how much this impacted the choice of estimation strategy. The goal of the experiment is to improve our understanding of the robustness of the observed strategy preferences. If a software developer’s strategy preference is stable and based strongly on previous experience, it is less likely that priming effects will be observed. If, on the other hand, there is a large “random” element in choice of estimation strategy, the priming effects may be substantial.

8.1 Design

The participants in Experiment 4 were 111 software developers who were attending a presentation on estimation. A few (less than 10) of the software developers had participated in Experiment 1 or 2. However, the participation in Experiments 1 or 2 had taken place over six months previously. That being so, we thought it unlikely that there would be any learning effect from these experiments.

The participants:

1. were randomly divided into two groups (Group Analogy and Group Average),
2. received the estimation material
3. completed the tasks (the priming part of the experiment):
 - a. Group Analogy participants completed three tasks in which they were asked to provide the closest analogies: (i) the Norwegian village closest to Geilo [a Norwegian winter sport village] in number of inhabitants, (ii) the profession perceived to be most similar to software developers (professions selected from a list of options), and (iii) the programming language most similar to Java with respect to productivity measured as lines of code per work-hours (programming languages selected from a list of options).

- b. Group Average participants completed three tasks in which they estimated average values: (i) the average height of 18-year-old Norwegians, (ii) the average number of emails received by Norwegian software developers per day, and (iii) the average lines of Java code of acceptable quality written by Norwegian software developers per day.
4. completed the same estimation task as in Phase 1 of Experiments 1 and 2; see Section 5.1 for a description.

If the priming tasks had an effect on the strategy preference, i.e., if the strategy preference was not very robust, we would expect (i) the proportion of developers who selected C1An would be larger in Group Analogy than in Group Average, due to easier access to analogy-based strategies, and (ii) the proportion of developers who selected AggClass and AggAll would be larger in Group Average than in Group Analogy, due to easier access to aggregation-based strategies.

8.2 Results

We found clear differences in the use of strategies, dependent on the priming tasks, as displayed in Figure 4 and Table 12. The median difference in productivity of Group Analogy and Average was 10 work-hours per use case. As before, the vertical lines indicate the estimates that we expected from a pure use of an estimation strategy (C1An 30; AggClass 70; AggAll 80). Notice that the proportion of developers who chose AggClass is greater than in Experiments 1 and 2. A closer analysis of the responses suggests that the very high preference of the AggClass estimation strategy was only among those from the same company. This may mean that there were company-specific reasons for the strong preference of AggClass.

Figure 4: Productivity Histogram of Groups A and B

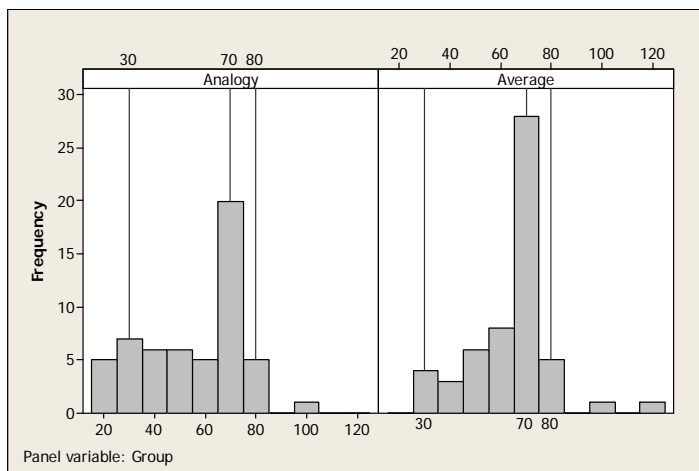


Table 12: Strategy Selection

Strategy	Group Average	Group Analogy
CIAn	7	18
AggClass	42	31
AggAll	7	6

We asked several (about twenty) of the participants informally about whether they felt that they were affected by the previous tasks (the priming tasks). None of them felt that this was the case.

We believe that the results of Experiment 4 suggest that strategy choices are not necessarily deeply rooted in previous experience, but may also be based on unconscious processes that are affected by various factors, such as the strategy that was applied most recently, i.e., the most “accessible” strategy. Similar accessibility results, although not related to the choice of problem solving strategy as far as we know, are reported in several other studies; see (Mussweiler 2003) for an overview.

9 Discussion

To some extent, the results of our experiments may reflect a general process for the selection and change of strategy, such that we initially choose an estimation strategy based on previous experience, values, and strategy accessibility and then update the strategy when the motivation is sufficiently strong. This motivation may be a result of large estimation errors when following the current strategy and/or a comparison of aggregated performance of the strategies included in our repertoire of strategies, e.g., judgment-based strategies with acceptable cost-benefit. In many ways, this is a rational process of strategy selection and change. However, it may have certain unfortunate implications, such as:

- Some software developers will sometimes continue to use an estimation strategy that is less accurate than another one when they do not experience a large estimation error following the preferred strategy. If, in addition, there are readily available external factors that can be blamed for having caused the estimation errors, such as requirement changes or external, unexpected events, it may even be possible to continue using the initially preferred strategy when large estimation errors are experienced.
- The initial strategy preference may be based on previous experience that is only weakly relevant for the current estimation context. Given that the software developers may not be aware of the source of the preference, as is suggested by Experiment 4, the validity of the preferred strategy may be difficult to assess.

At this very early stage of our understanding of the processes that software developers use when selecting and changing estimation strategies, it is difficult to offer recommendations for improving processes of judgment-based effort estimation. Nevertheless, the above limitations of current processes of selecting estimation strategies may provide some support in favor of training the software developers in the use of more analytic, explicit selection processes, e.g., processes that emphasize historical accuracy more than estimation surprises. However, more studies need to be conducted before we can be confident about whether or not a change to more analytical processes of selection would improve estimation accuracy.

10. Summary and Conclusions

This paper describes four experiments that aim at achieving a better understanding of the processes and preferences that lead to the selection and change of estimation strategy in judgment-based effort estimation.

Our preliminary answers on our three research questions stated in Section 4, based on the experiments from the results, are:

- There is a large individual variation in selection of strategies in situations where there is no information clearly in favor of one particular strategy (RQ1).
- We find evidence to support the contention that the following three strategy selection factors play important roles in situations where relevant, historical data are available: i) individual strategy preference, ii) estimation surprise, and iii) aggregated information that compares the previous accuracy of candidate estimation strategies in a particular estimation context. In the examined context, we found that estimation surprise was frequently a better indicator of strategy change than the historical accuracy of an estimation strategy (RQ2).
- The estimation strategy preferences may not be very strong, at least in situations like the one we have studied, and easily impacted by increased “accessibility” through use of a strategy on an unrelated task, immediately before (RQ3).

The large individual variance in the selection of an estimation strategy and the differences in weighting of the factors that drive changes of estimation strategy may be two reasons for the observed large variance and inconsistency in judgment-based effort estimation; see for example (Jørgensen and Carelius 2004; Jørgensen and Sjøberg 2004; Grimstad and Jørgensen 2007). The use of different estimation strategies means that we should expect different effort estimates even when the project to be estimated and/or the estimation context are the same.

Our results are derived from experiments in which several of the elements were deliberately designed to be artificial, e.g., the limited information about each project. While these elements enabled us to identify the estimation strategies used by the software developers to some extent, they

also mean that we should be very careful when generalizing to other types of effort estimation situations than the one we studied. In effort estimation situations in which a greater amount of information about the project to be estimated and a smaller amount of historical data is available, and more time is spent on effort estimation, strategies other than the ones we examined may dominate. In spite of these limitations, we believe that the examined strategies will form part of most estimation work. A central reason for this is that it is hard to imagine effort estimation that is not based on historical data. "Representativeness" has been found to be an important approach for the identification of relevant historical data; see our discussion in Section 2.1. "Representativeness" has to be based on either specific or aggregated historical data, or a combination of them. We believe that the strategies examined in our experiments reflect natural implementations of specific and aggregated historical data in effort estimation contexts.

We have been unable to find other studies on the selection and change of judgment-based effort estimation strategies. This lack of studies limits our ability to improve judgment-based effort estimation, which is the dominant approach to estimation in the software industry. However, we hope that our preliminary results may be of some use:

- to increase the understanding of judgment-based effort estimation, e.g., to supply potential reasons for the observed inconsistency of software developers' effort estimates, and,
- to provide input to further studies, e.g., as input to follow-up studies that will evaluate the robustness of our results in effort estimation situations in which a greater amount of information about the projects is available and as input to studies that propose improved processes of judgment-based effort estimation.

References:

- Aranda, J. and S. M. Easterbrook (2005). Anchoring and Adjustment in Software Estimation. European Software Engineering Conference / ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'05), Lisbon, Portugal.
- Berger, J. O. (1985). Statistical Decision Theory and Bayesian Analysis, Springer-Verlag.
- Briand, L. C., K. El Emam, D. Surmann, I. Wiczorek and K. D. Maxwell (1999). An assessment and comparison of common software cost estimation modeling techniques. International Conference on Software Engineering, Los Angeles, USA, ACM, New York.
- Cacioppi, J. T. and R. E. Petty (1982). "The need for cognition." Journal of Personality and Social Psychology **42**(1): 116-131.
- Evans, J. S. t. (2003). "In two minds: dual-process accounts of reasoning." Trends in cognitive sciences **7**(10): 454-459.

- Gray, A., S. G. MacDonell and M. J. Shepperd (1999). Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment. IEEE 6th International Metrics Symposium, Boca Raton, Fl., IEEE Computer Society.
- Grimstad, S. and M. Jørgensen (2007). "Inconsistency in Expert Judgment-based Estimates of Software Development Effort." Journal of Systems and Software **80**(11): 1770-1777.
- Hammond, K. R. (1996). Human judgement and social policy: Irreducible uncertainty, inevitable error, unavoidable injustice. New York, Oxford University Press.
- Hammond, K. R., R. M. Hamm, J. Grassia and T. Pearson (1987). "Direct comparison of the efficacy of intuitive and analytical cognition in expert judgment." IEEE Transactions on Systems, Man, and Cybernetics **17**(5): 753-770.
- Harvey, N. (2007). "Use of heuristics: Insights from forecasting research." Thinking & Reasoning **13**(1): 5-24.
- Hogarth, R. M. (2005). Deciding Analytically or Trusting your Intuition? The Advantages and Disadvantages of Analytic and Intuitive Thought. The routines of decision making. T. Betsch and S. Haberstroh. Mahwah, NJ, Erlbaum: 67-82.
- Jørgensen, M. (1995). "An empirical study of software maintenance tasks." Software Maintenance: Research and Practice **7**: 27-48.
- Jørgensen, M. (2004). "A review of studies on expert estimation of software development effort." Journal of Systems and Software **70**(1-2): 37-60.
- Jørgensen, M. (2005). The "Magic Step" of Judgment-Based Software Effort Estimation. International Conference on Cognitive Economics Sofia, Bulgaria.
- Jørgensen, M. (2007). A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation. 1st International Workshop on Software Productivity Analysis and Cost Estimation, Nagoya, Information Processing Society of Japan.
- Jørgensen, M. (2007). "Estimation of Software Development Work Effort: Evidence on Expert Judgment and Formal Models." International Journal of Forecasting **23**(3): 449-462.
- Jørgensen, M. (2007). Individual Differences in How Much People are Affected by Irrelevant and Misleading Information. Second European Conference on Cognitive Science, Delphi, Greece, Hellenic Cognitive Science Society.
- Jørgensen, M. and G. Carelius (2004). "An empirical study of software project bidding." IEEE Transactions on Software Engineering **30**(12): 953-969.
- Jørgensen, M. and S. Grimstad (2008). "How to Avoid Impact from Irrelevant and Misleading Information When Estimating Software Development Effort." IEEE Software **May/June**: 78-83.

- Jørgensen, M. and T. M. Gruschke (2005). Industrial Use of Formal Software Cost Estimation Models: Expert Estimation in Disguise? EASE, Keele, UK.
- Jørgensen, M. and T. M. Gruschke (2008). "The Impact of Lessons Learned Sessions on Effort Estimation and Uncertainty Assessments." submitted to IEEE Transactions on Software Engineering.
- Jørgensen, M. and D. I. K. Sjøberg (2001). "Impact of effort estimates on software project work." Information and Software Technology **43**(15): 939-948.
- Jørgensen, M. and D. I. K. Sjøberg (2004). "The impact of customer expectation on software development effort estimates." International Journal of Project Management **22**: 317-325.
- Kahneman, D., P. Slovic and A. Tversky (1982). Judgment under uncertainty: Heuristics and biases. Cambridge, United Kingdom, Cambridge University Press.
- Kahneman, D. and A. Tversky (1973). "On the psychology of prediction." Psychological Review **80**(4): 237-251.
- Levine, R. L., M. E. Huneke and J. D. Jasper (2000). "Information Processing at Successive Stages of Decision Making: Need for Cognition and Inclusion–Exclusion Effects." Organizational Behavior and Human Decision Processes **82**(2): 171-193.
- Miyazaki, Y., M. Terakado, K. Ozaki and H. Nozaki (1994). "Robust regression for developing software estimation models." Journal of Systems and Software **27**(1): 3-16.
- Mussweiler, T. (2003). "Comparison Processes in Social Judgment: Mechanisms and Consequences." Psychological Review **110**(3): 472-489.
- Niessink, F. and H. van Vliet (1997). Predicting maintenance effort with function points. International Conference on Software Maintenance, Bari, Italy, IEEE Comput. Soc, Los Alamitos, CA, USA.
- Northcraft, G. B. and M. A. Neale (1987). "Expert, amateurs, and real estate: An anchoring-and-adjustment perspective on property pricing decisions." Organizational Behavior and Human Decision Processes **39**: 228-241.
- Osman, M. (2004). "An evaluation of dual-process theories of reasoning." Psychonomic Bulletin & Review **11**(6): 988-1010.
- Rieskamp, J. and P. E. Otto (2006). "SSL: A theory of how people learn to select strategies." Journal of experimental psychology: General **135**(2): 207-236.
- Roberts, M. J. (2000). Individual differences in reasoning strategies: a problem to solve or an opportunity to seize? Deductive reasoning and strategies. W. Schaeken, B. D. Vooght, A. Vandierendonck and G. D'Ydewalle. Mahwah, Lawrence Erlbaum: 23-48.
- Roberts, M. J. and E. J. Newton (2001). "Understanding strategy selection." International journal of human-computer studies **54**: 137-154.

- Rush, C. and R. Roy (2001). "Expert judgement in cost estimating: modelling the reasoning process." Concurrent Engineering: Research and Applications **9**(4): 271 - 284.
- Schacter, D. L., I. G. Dobbins and D. M. Schnyer (2004). "Specificity of priming: A cognitive neuroscience perspective." Neuroscience **5**: 853-862.
- Shepperd, M., M. Cartwright and G. Kadoda (2000). "On building prediction systems for software engineers." Empirical Software Engineering **5**(3): 175-182.
- Shepperd, M. and G. Kadoda (2001). "Comparing software prediction techniques using simulation." IEEE Transactions on Software Engineering **27**(11): 1014-1022.
- Shynkaruk, J. M. and V. A. Thomson (2006). "Confidence and accuracy in deductive reasoning." Memory & cognition **34**(3): 619-632.
- Silverman, B. G. (1985). "Expert intuition and ill-structured problem solving." IEEE Transactions on Engineering Management **EM-32**(1): 29-33.
- Simon, H. A. (1957). Models of Man, Social and Rational: Mathematical Essays on Rational Human Behavior in a Social Setting. New York, Wiley.
- Wills, A. J., A. Lavric, G. S. Croft and T. I. Hodgson (2007). "Predictive learning, prediction errors, and attention: Evidence from event-related potentials and eye tracking." Journal of Cognitive Neuroscience **19**(5): 843-854.