Using planning poker for combining expert estimates in software projects

ARTICLE *in* JOURNAL OF SYSTEMS AND SOFTWARE · DECEMBER 2008

Impact Factor: 1.35 · DOI: 10.1016/j.jss.2008.03.058 · Source: DBLP

3 AUTHORS, INCLUDING:

Kjetil Moløkken-Østvold
PwC
24 PUBLICATIONS **758** CITATIONS
SEE PROFILE

Hans Christian Benestad
Simula Research Laboratory
15 PUBLICATIONS **111** CITATIONS
SEE PROFILE

The Journal of Systems and Software 81 (2008) 2106–2117

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Using planning poker for combining expert estimates in software projects

Kjetil Moløkken-Østvold [a,b,*], Nils Christian Haugen [a], Hans Christian Benestad [a]

[a] Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway
[b] Conceptos IT Development, P.O. Box 279, 1326 Lysaker, Norway

ARTICLE INFO

Article history:
Received 31 July 2007
Received in revised form 11 January 2008
Accepted 26 March 2008
Available online 23 April 2008

Keywords:
Software estimation
Planning poker
Group processes
Project management

ABSTRACT

When producing estimates in software projects, expert opinions are frequently combined. However, it is poorly understood whether, when, and how to combine expert estimates. In order to study the effects of a combination technique called planning poker, the technique was introduced in a software project for half of the tasks. The tasks estimated with planning poker provided: (1) group consensus estimates that were less optimistic than the statistical combination (mean) of individual estimates for the same tasks, and (2) group consensus estimates that were more accurate than the statistical combination of individual estimates for the same tasks. For tasks in the same project, individual experts who estimated a set of control tasks achieved estimation accuracy similar to that achieved by estimators who estimated tasks using planning poker. Moreover, for both planning poker and the control group, measures of the median estimation bias indicated that both groups had unbiased estimates, because the typical estimated task was perfectly on target. A code analysis revealed that for tasks estimated with planning poker, more effort was expended due to the complexity of the changes to be made, possibly caused by the information provided in group discussions.

1. Introduction

In the software industry, various techniques are used to combine estimates. One of the most recent additions is *planning poker*, introduced by Grenning (2002) and also described in a popular textbook on agile estimating and planning by Cohn (2005). There exist few empirical studies on the combining of estimates in software engineering, but there are some indications that combination may reduce the bias towards optimism in estimates (Moløkken-Østvold and Jørgensen, 2004).

However, plenty of evidence from other research communities details the possible hazards of group processes (Brown, 2000). For example, some recent papers published in psychology and forecasting emphasize the problems of decision making in groups (Armstrong, 2006; Buehler et al., 2005). Bueler et al. found that (a) both individual and group predictions had an optimistic bias, (b) group discussion increased individual biases, and (c) this increase of bias in groups was mediated by the participants' focus on factors that promote the successful completion of tasks (Buehler et al., 2005). Scott Armstrong states that he has been unable to obtain evidence that supports the use of face-to-face groups in decision making (Armstrong, 2006).

These recent observations are in line with many previous classic studies on decision making in groups; individuals are inherently optimistic and this optimistic bias is increased by group interaction (Brown, 2000; Aronson et al., 1999; Atkinson et al., 1996).

In contrast to the foregoing, our studies on software expert estimates have found that individuals are, in general, biased towards optimism, but that this bias can actually be *reduced* by group discussions (Moløkken-Østvold and Jørgensen, 2004).

The explanation for this apparent disparity may be that there are differences between a typical software estimation task and the tasks studied in other research areas. First, other studies frequently use ad hoc groups (e.g. Buehler et al., 2005), whereas software estimation usually involves professionals who are accustomed to collaborating with each other and are motivated to perform in a professional manner (Moløkken-Østvold and Jørgensen, 2004). Second, other studies tend to use tasks of which kind the participants might have little experience (Brown, 2000), whereas in software projects the participants are usually experienced. Third, another often-reported problem is that laboratory studies tend to investigate hypothetical task and/or outcomes (Brown, 2000), and not real executed tasks with a recorded outcome.

It might be that estimating software projects is a type of task that it is, for some reason, sensible to discuss in groups. However, it might also be that previous studies in software engineering have had methodological shortcomings.

The purpose of this study was to (1) explore the group processes that may occur when planning poker is used to estimate tasks, and (2) compare planning poker estimates with existing individual

* Corresponding author. Address: Conceptos IT Development, P.O. Box 279, 1326 Lysaker, Norway.
E-mail addresses: kjetilmo@simula.no (K. Moløkken-Østvold), nch@simula.no (N.C. Haugen), benestad@simula.no (H.C. Benestad).

0164-1212/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.03.058

estimation methods. Section 2 introduces group estimation. In Sections 3 and 4, respectively, research questions and methods are described. The results are presented in Section 5 and discussed in Section 6. Section 7 concludes.

2. Combining estimates in groups

Group estimation has not been widely studied in a software engineering context. In fact, a recent review (Moløkken-Østvold and Jørgensen, 2004) of the leading journals in the systems and software engineering field did not find a single paper that described empirical studies of group estimates in an industrial context. Since that review, at least two studies have been published, one of which compared individual expert estimates (combined in statistical groups) with an unstructured group estimation method (Moløkken-Østvold and Jørgensen, 2004) and the other of which compared unstructured group estimates with planning poker (Haugen, 2006). In addition, the combining of estimates has been studied in student tasks (Passing and Shepperd, 2003; Höst and Wohlin, 1998).

Various techniques can be used to combine estimates. A simplified overview of six of the most common techniques, including what we perceive to be central properties, is displayed in Table 1.

Structure describes the level of formality, amount of learning requirements, and degree of rigidity associated with the technique. *Anonymity* describes whether the estimators are anonymous to each other. *Interaction* describes whether, and if so to what extent, the estimators interact with each other. *Overhead* describes the typical extra amount of effort spent on estimating each project or task.

Perhaps the most well-known technique for combination is the Delphi technique (Helmer, 1966), which was devised by the RAND corporation in the 1950s (Rowe and Wright, 1999). The Delphi technique does not involve face-to-face discussions, but anonymous expert interaction through several iterations, supervised by a moderator until a majority position is attained. In addition to *anonymity*, the method needs to include *iterations*, *controlled feedback* and *statistical aggregation of responses* for it to be implemented properly (Rowe and Wright, 1999).

It is claimed that even though the technique has been used widely, actual scientific studies of the techniques' merits are sparse and often conducted inappropriately (Rowe and Wright, 1999; Powell, 2003). However, even though reviews advise caution, there is evidence that the Delphi technique outperforms statistical groups and unstructured interacting groups (Rowe and Wright, 1999) and that it is a sound method for harnessing the opinions of a diverse group (Powell, 2003). However, there is no conclusive evidence that Delphi outperforms other structured group combination techniques.

We have found no empirical research on the accuracy of Delphi in a software engineering context. However, it is frequently recommended in papers on software management, e.g. Fairley, 2002.

The Wideband Delphi technique is a modification of the Delphi technique and includes more group interaction than Delphi (Boehm, 1981). As in the Delphi technique, there is a moderator, who supervises the process and collects estimates. However, in this approach the experts meet for group discussions both prior to, and during, the estimation iterations.

Table 1
Overview of some common combination techniques

Method	Structure	Anonymity	Interaction	Overhead
Delphi	Heavy	Yes	No	Major
Wideband Delphi	Moderate	Limited	Limited	Moderate
Planning Poker	Light	No	Yes	Limited
Unstructured groups	Light	No	Yes	Limited
Statistical groups	Light	Yes	No	Limited
Decision markets	Heavy	Yes	No	Moderate

The Wideband Delphi technique is very similar to the Nominal Group technique, which is also know as the estimate-talk-estimate technique (Van de Ven and Delbecq, 1971). Due to its similarities to the Wideband Delphi technique, the Nominal Group technique is neither presented nor discussed in this paper.

Wideband Delphi has been proposed as an estimation method in books Boehm (1981), and papers on software metrics (Fenton, 1995) and software process improvement (Humphrey, 1990). To the best of our knowledge, the Wideband Delphi technique has not been studied empirically.

The planning poker technique is relatively new. It is a lightweight technique, with face-to-face interaction and discussions. In short, the steps of the technique, as originally described by Grenning, are "*The customer reads a story. There is a discussion clarifying the story as necessary. Each programmer writes their estimate on a note card without discussing their estimate. Anticipation builds. Once all programmers have written their estimate, turn over all the cards. If there is agreement, great, no discussion is necessary, record the estimate and move on to the next story. If there is disagreement in the estimates, the team can then discuss their different estimates and try to get to consensus* (Grenning, 2002)". By *story*, Grenning means a *user story*. A user story is a software system requirement that is formulated as one or two sentences in the everyday language of the user. The technique, and how it was adopted to the project studied, will be described in greater detail in Section 4.

Being a relatively new technique, planning poker has, as far as we are aware, been the subject of only one published empirical study (Haugen, 2006). In that study, planning poker was compared to unstructured group estimation. It was found that for familiar tasks, the planning poker technique produced more accurate estimates than unstructured combination, whereas the opposite was found for unfamiliar tasks.

Unstructured group combination is, as the name implies, basically discussions with a group decision being made at the end. Depending on needs, individuals can derive their own estimates before the discussion.

A review of the literature on forecasting (Rowe and Wright, 2001) suggests that unstructured groups were, on average, outperformed by Delphi-groups. However, the review also found that there are tasks for which unstructured groups are better suited. In some situations, it is possible that an unstructured group can outperform a Delphi group if the motivation of, and information sharing among, the participants is adequate (Rowe and Wright, 2001).

In a previous study on software estimation (Moløkken-Østvold and Jørgensen, 2004), we found that group estimates made after an unstructured discussion were less optimistic and more realistic than individual estimates derived prior to the group discussion and combined in a statistical group. The main sources of this decrease in optimism seemed to be the identification of additional activities and an awareness that activities may be more complicated than was initially thought.

Note that study used an unstructured technique that involved prior individual estimates. Often, companies use an unstructured combination where experts meet to provide consensus estimates, without having previously made their individual estimates. This latter procedure is perhaps more susceptible to peer-pressure than when individual estimates have been derived initially.

In a statistical group, there is no interaction between the group members. They are a group only in the sense that their individual estimates are combined statistically.

When considering how to combine estimates given by several individuals into an estimate, well-known statistical methods can be used. Computing the mean or median of the different individual estimates will give us one estimate that is based on multiple estimates.

Jørgensen claims that taking a simple average often works as the best method for combining estimates (Jørgensen, 2005).

A decision market is a technique for combining opinions that can also be used in estimation. Hanson provides the following definition: *"Decision markets are (markets) designed primarily for the purpose of using the information in market values to make decisions* (Hanson, 1999)".

A decision market can be set up like a stock market, with decisions being substituted for stocks. Traders are invited to invest money in the alternative, represented by stocks (decisions), that they think will be the eventual outcome. A trader holding a stock (decision) that becomes the actual outcome receives a fixed amount of money, prize or similar. Through the dynamics of a market, this results in higher stock (decision) prices for the alternatives that most people think will be the outcome, which creates a likelihood distribution for the different outcomes.

According to Surowiecki, such a market is wise because it aggregates the opinions of traders. A market may be especially powerful if the traders are diverse in their backgrounds, independent of each other, and have local knowledge (Surowiecki, 2004).

Inspired by Surowiecki, Berndt, Jones et al. advocate the use of decision markets in software effort estimation (Berndt et al., 2006). They stress that by allowing all project stakeholders to participate in the decision market, one ensures diversity in the input to the estimation process and aggregates the knowledge from all the project stakeholders. According to Berndt, Jones et al., another positive feature of decision markets is that the different traders can apply whatever estimation technique they like, thus enabling a combination of different estimation techniques.

A decision market is, as is Delphi, a way of aggregating different opinions without face-to-face meetings. Like Delphi, a decision market seeks to preempt the social and political problems caused by the use of interacting groups, while at the same time utilizing the increase in knowledge that using groups offers. An important factor is that the participants can receive (continuous) feedback on their own opinion compared to others.

The main difference between Delphi and decision markets is the way in which the knowledge and opinions of the group members are aggregated.

We have not managed to find any empirical research on the use of decision markets for software estimates. However, a recent paper by Berndt et al. (2006) describes an ongoing study.

Studies on the combining of estimates for student tasks have shown some positive effects, both when combining estimates statistically (Höst and Wohlin, 1998) and in face-to-face discussions (Passing and Shepperd, 2003).

To summarize, the strategy of combining estimates for groups in general, and for software estimation in particular, is far from understood.

In addition, some studies, e.g., by Buehler et al. (2005), specify some limitations that may reduce the strategy's applicability to real life problems, mainly that the groups studied consisted of individuals who were unfamiliar with each other.

It is also important to note that findings may vary in their applicability as task characteristics, motivational factors, social relations, and communication structure differ. In general, when reviewing studies on group processes, it is also important to differentiate between studies in which the participants cannot influence the outcome and those in which they can.

3. Research questions

It is possible that typical software estimation tasks are suitable for group combination, such as planning poker. In an estimation process, there may be several experts who contribute different project experiences and knowledge. Such experiences can be shared more easily in a face-to-face group, as with planning poker, than through a moderator, as with the Delphi technique. In addition, face-to-face interaction may make the participants more committed to the decisions.

We wanted to further explore whether, as found in a previous study on group estimation (Moløkken-Østvold and Jørgensen, 2004), optimism could be reduced by group discussions. From this, we derived the following research question:

RQ1: Are group consensus estimates less optimistic than the statistical combination of individual expert estimates?

We define optimism of estimates in the relative sense, and irrespective of accuracy. We deem one estimate to be more optimistic than another if and only if it states that it will take less time to complete a task than the other estimate, i.e., an estimate of 4 h is more optimistic than an estimate of 5 h.

Any observation of reduced optimism would indicate a *choice shift*, defined by Zuber et al. (1992) as the difference between the arithmetic average (mean) of individual decisions and the group consensus decision. Such an observation of reduced optimism would be contrary to that which is typically reported from other research areas, where the choice shift is generally in the direction of increased risk willingness and optimism (Buehler et al., 2005).

It is important to note that our study does not merely confirm or undermine the results of the previous study on group estimation (Moløkken-Østvold and Jørgensen, 2004). It also generates a new result, because the subjects used planning poker rather than the unstructured discussion used in the previous study.

Even if a reduction in optimism were observed, it would not necessarily guarantee that estimates would be more accurate, because some or all of the individual estimates might already be biased towards pessimism. Thus, we also wanted to investigate whether group consensus estimates made after a discussion are more accurate than mean individual estimates. This concern generated the following research question:

RQ2: Are group consensus estimates more accurate than the statistical combination of individual expert estimates?

The previous empirical study on planning poker compared planning poker with an unstructured method for combining estimates in groups (Haugen, 2006). Therefore, we wanted to compare the estimates that were derived by using planning poker to a series of estimates that were derived by individual experts, and not subject to subsequent group discussions. This generated the following research question:

RQ3: Are group consensus estimates more accurate than the existing individual estimation method?

In addition to possibly influencing estimation accuracy, the introduction of group estimation might lead to changes in how the developers work. Such changes might include differences in the amount of total effort spent on the estimation phase, or effort spent on restructuring code during implementation. The final research question to explore is therefore

RQ4: Does the introduction of a group technique for estimation affect other aspects of the developers' work, when compared to the individual estimation method?

The sizes of changes have been shown to be fundamental in explaining change effort variations; see, e.g., Graves and Mockus, 1998. It is therefore necessary to explore:

RQ4A. Are there differences between the planning poker tasks and the control group tasks that are related to the size of the changes?

A larger change size for the planning poker tasks may, at least partly, explain any differences between the groups in actual effort, and provide an intermediate link for the causal analysis.

The other subquestion to investigate is

RQ4B. Are there differences between the planning poker tasks and the control group tasks that are related to the complexity of the changes?

Conclusions drawn from the analysis for question 4B are tentative, because only a subset of possible factors was investigated. However, the analysis can provide partial evidence and insight that can be useful in a wider causal analysis in combination with the other study results. Thus, it is of interest to assess any difference in effort after controlling for possible differences in change size and complexity.

Research questions 1 and 2 concern *intragroup* differences, while research questions 3, 4A and 4B concern *intergroup* differences.

4. Research method

The research method was designed to address some of the issues pertaining to validity that arose in our previous studies (Moløkken-Østvold and Jørgensen, 2004; Haugen, 2006).

The main limitation of the previous study of individual estimates (combined in statistical groups) followed by unstructured group combination was that the groups of professionals did not themselves implement the project they estimated (Moløkken-Østvold and Jørgensen, 2004). This was done by a separate team; thus, the estimators did not estimate their own work. Therefore, from the perspective of the estimators in that study, there was a hypothetical outcome. However, as the project was actually implemented, it was possible to discern estimates that were clearly optimistic or pessimistic.

The previous study on planning poker was limited to one team (Haugen, 2006). Another limitation was an unknown effect of increased system experience, because there was no randomization of tasks to the different methods (unstructured group estimation vs. planning poker). In addition, the study compared two different group estimation methods (planning poker vs. unstructured groups), and did not compare group estimation with individual estimation.

In the study reported herein, we wanted a design that could both measure any shift in choices among the planning poker tasks and compare planning poker with a control group of tasks estimated with the existing individual estimation method. In addition, it was important that the design allowed for the comparison of estimates with the actual effort for all tasks.

We also wanted to perform an analysis of code following the completion of the tasks, to explore any possible differences related to the size or complexity of the changes. Finally, we wanted to conduct face-to-face interviews with the participants, in order to further explore and explain possible findings.

A simplified overview of the study design is presented in Table 2.

4.1. The company and project studied

The company studied is a medium-size Norwegian software company that delivers custom-made solutions to various private and public clients. The project team studied had been working for a large public client for several months at the start of the study and was using Scrum (http://www.controlchaos.com/) as the project methodology. The project team estimates the tasks to be performed in the upcoming *sprint* (Scrum terminology for the next period). In the team studied, this happens once each fortnight, and about 15–20 tasks are selected for each sprint. From four to six team members participate in each sprint, depending on the demands of the tasks.

All project participants in the study were guaranteed anonymity and assured that no results regarding performance could be traced.

4.2. The estimation methods studied

The existing estimation method of the project was that the tasks were estimated individually by the team member responsible for the part of the system that would be affected by the change. Changes requested by the client were analyzed, estimated and recorded (Step A; Table 2) in their task tracking system, Jira (http://www.atlassian.com/software/jira/). The tasks were of varying size and character, ranging from two-hour bug fixes to three-day analyses of larger changes, and were relatively well defined in the task tracking system, which used text and screenshots. All members of the team participated in estimating tasks. This was done individually and estimates were not revealed to other team members. This method was used in the control group in our study.

For each sprint, half of the tasks were to be re-estimated with a variation of planning poker, while the initial estimate was retained for the other half (Step C). The tasks were assigned randomly to either the planning poker or the control group (Step B).

Before the study, the team was given an introduction to planning poker by the company's chief scientist. The estimation method for the planning poker condition was employed with the following steps in sequence for each task:

1. The task was presented to the team (Step D) by the developer who registered the task in the task tracking system. The initial estimate was not revealed to other team members and was discarded (Step C).
2. The task was discussed briefly by the team (Step E), to ensure that everybody had the same interpretation before estimates were made.
3. The team members then estimated, individually, the most likely effort needed to perform the task specified (Step F). The estimate was given in work-hours.
4. All team members revealed their estimates simultaneously (Step G):
    a. If any estimates were larger than 18 h, there was a brief discussion of how to break the task down into subtasks. A full day of work was deemed to be 6 h. From previous experience, the team felt that estimates larger than 18 h (i.e. three days) were less accurate. Therefore, they decided to split tasks above this size.
    b. Those with the lowest and highest estimates had to justify their estimates. A brief debate followed (Step H). The debate was led by the company's chief scientist for the first sprint; thereafter, the team worked on its own.
    c. If a consensus was reached on an estimate, this was recorded (Step I) and the team moved on to the next task (Step C). If no consensus could be reached, the members revised their estimates and participated in a new individual estimation round for that particular task.
5. After the task had been performed, The developer who performed the task recorded the actual effort expended, together with his or her initials (Step K).

Table 2
Study overview

	Planning poker	Control group
A	Tasks requested by client entered into the task tracking system and given an initial estimate	
B	Tasks assigned randomly to planning poker or control group	
C	Initial estimate discarded	Initial estimate kept
D	Task presented to team	N/A
E	Task discussed briefly	N/A
F	Individual estimates derived	N/A
G	Individual estimates revealed	N/A
H	Estimates discussed	N/A
I	Consensus estimate derived by group	Initial estimate used as estimate
J	Task performed	
K	Actual effort recorded	
L	Source code analyzed	
M	Participants interviewed	

Note that this method differs somewhat from the description given by Grenning (2002). For example, planning poker was, in our case, used for task estimation, and not estimation of user stories (Cohn, 2005) or features (for which use it is most commonly recommended).

After all the tasks had been performed, the code for solutions in both study groups was analysed (Step L) and the participants were interviewed to get their opinions (Step M).

4.3. Calculation of estimation accuracy

To calculate estimation accuracy, we employed the BRE (balanced relative error), because it is a more balanced measure than the MRE (Miyazaki et al., 1991). It is calculated as:

$$\mathrm{BRE} = \frac{|x-y|}{\min(x,y)}, \quad x = \text{actual and } y = \text{estimate}$$

In order to measure whether there was a bias towards optimism or pessimism, the BREbias was calculated, because this measures both the size *and* the direction of the estimation error:

$$\mathrm{BREbias} = \frac{(x-y)}{\min(x,y)}, \quad x = \text{actual and } y = \text{estimate}$$

To measure the size of any difference in mean values, we used Cohen's size of effect measure (d) (Cohen, 1969), where

$$d = \frac{\overline{\text{sample1}} - \overline{\text{sample2}}}{\text{pooledStdDev}}$$

Here, pooledStdDev denotes the *pooled standard deviation*, a method for assessing the true standard deviation of different samples.

4.4. Code analysis

Checkins to the code repository, Subversion (http://subversion.tigris.org), were tagged by the developers with a task identifier. Hence, we were able to retrieve the exact state of the application before and after each task, and quantitative measures of changes could be extracted. By studying the changes to the application code associated with the individual tasks, we were better prepared to investigate and discuss whether, and how, the estimation method may have influenced how effort was spent for each task.

More specifically, the code analysis sought to compare the amount and complexity of code that was added, changed, and removed during the tasks. This allowed a more focused root cause analysis of possible differences in change effort. Research question 4A was investigated by inspecting mean and median values for change size and performing a Kruskal–Wallis test (Wonnacott and Wonnacott, 1990) on the medians.

For both questions, regression models of change effort vs. measures of change size and change complexity were used to gain insight into factors that explain change effort. When applying these models and measures, we used the same statistical framework as, and similar procedures to, those discussed and used by Graves and Mockus (1998), who performed similar analyses of change effort. In brief, the framework utilizes Generalized Linear Models using change effort (measured in work-hours) as a dependent variable, measures of possibly influential factors as covariates, and a log link. It assumes Poisson distributed errors, and allows a free scale parameter to adjust for possible overdispersion (c.f., Myers et al., 2002).

Although artifacts such as binaries, design models, and build scripts were present in the code repository, only *source code* was considered for this analysis. Source code included files for Java, Java Server Pages, the eXtensible Stylesheet Language, XML, and XML Schema Definitions.

Added, deleted and changed lines were measured by processing the side-by-side ($-y$ option) output of the standard Linux program diff (c.f., Hunt and McIlroy, 1976, MacKenzie et al., 2003). Frequently used measure of change size is the sum of these measures (SIZE1); see e.g., Jørgensen, 1995. Graves and Mockus (1998) evaluated several size measures and found that the number of file checkins to the code repository (SIZE2) best explained change effort. We constructed a third measure, the number of changed segments of code (SIZE3). A changed segment of code is a set of consecutive lines of code, where all lines were either added, changed or deleted. The measures from the code repository were extracted automatically. Their definitions are provided in Table 3.

In order to select the most appropriate size measure among the three candidates, the deviance of the three regression models of the type described above was compared. Lower deviance values indicate a better fit between model and the actual data (Myers et al., 2002). A model based on SIZE3 and a mathematical intercept value provided the best fit and SIZE3 was selected as the size measure to use for the analysis; see Table 4.

We used three types of measures of change complexity. These were hypothesized to explain possible change effort variations:

Table 3
Change measures

ADD	Number of source code lines added
CH	Number of source code lines changed
DEL	Number of source code lines deleted
SIZE1	ADD + CH + DEL
SIZE2	Number of file revisions checked in (deltas)
SIZE3	Number of changed segments of code
ACS	Number of control-flow statements added
DCS	Number of control-flow statements deleted
AOR	Number of out-of-class references added
DOR	Number of out-of-class references deleted
SZAFF	Mean number of source code lines in affected modules
isControl	Binary variable representing group membership. Value set to 1 if task was in control group, 0 if task estimated by PP

Table 4
Model fit of size measures

Variables	Coefficient	p-Value	Deviance
Intercept	2.05	<0.0001	188
SIZE1	0.000738	0.0487	
Intercept	2.03	<0.0001	180
SIZE2	0.0203	0.0177	
Intercept	1.95	<0.0001	154
SIZE3	0.00793	0.0009	

Measures of the size of the affected code, similar to SZAFF, have been found by other researchers to affect change effort significantly (Niessink and van Vliet, 1998; Arisholm, 2006).

Measures of the type of change, similar to ADD, DEL, CH were used by, e.g., Jørgensen, 1995. Measures of additions and deletions of structural attributes (ACS, DCS, AOR, DOR) are less common, but have been investigated at the file level by Fluri and Gall (2006). An out-of-class reference means that the measured class uses a method or attribute in another class. A control-flow statement changes the sequential flow of control. Hence, the measures are similar to the concepts of import coupling (Briand et al., 1999) and cyclomatic complexity (McCabe, 1976), but adapted to measuring complexity *change* at the task level.

4.5. Interviews

All project participants were interviewed individually on a range of issues. These interviews sought to (a) uncover background information regarding project priorities, (b) ask specific questions regarding the planning poker technique, and (c) determine the participants' perception of differences between the planning poker technique and the individual estimation technique.

The interviews were performed in person and followed a structured questionnaire.

5. Results

In total, 55 tasks were estimated and implemented, of which 24 were estimated with planning poker and 29 with the existing individual estimation method. Two tasks were deleted from the dataset due to suspicion of faulty registration in the database. A brief summary of important data is presented in Table 5. The first column represents the initial estimates (Step A; Table 2), the second the statistical combination of individual estimates for the planning poker tasks (Step F), the third the final estimates (consensus estimate for the planning poker tasks, individual estimate for the control group, Step I), the fourth column the actual effort (Step J), the fifth the estimation accuracy of planning poker tasks measured against the statistical combination of individual estimates, the sixth estimation accuracy against final estimates, and the final column contains estimation bias.

The first two research questions relate only to the 24 tasks that were estimated with planning poker. The third research question compares accuracy results from the 24 tasks estimated by using planning poker with the 29 tasks estimated by using the existing individual estimation method. The final research question examines possible differences in change size and complexity.

5.1. RQ1: Are group consensus estimates less optimistic than the statistical combination of individual expert estimates?

For the 24 tasks that were estimated using planning poker, the mean of the statistical combination of individual estimates before group discussion was 6.3 h (median 6.0 h). After group discussion, the mean consensus estimate was 7.1 h (median 6.0 h). An analysis was performed with a paired *t*-test, as suggested in similar research on choice shift (Liden et al., 1999). Since the research question suggests a direction of effect (group discussion reduces optimism), the paired *t*-test was one-sided. We provide the actual *p*-values, as suggested by Wonnacott and Wonnacott (1990), instead of predefining a significance level for rejection. The results are displayed in Table 6.

The analysis of possible choice shift on the estimates yielded a *p*-value of 0.04 and an effect-size of 0.16.

5.2. RQ2: Are group consensus estimates more accurate than the statistical combination of individual expert estimates?

On the basis of the estimates, we calculated the estimation accuracy measured in BRE. The mean BRE of the statistical combination of individual estimates before group discussion was 0.94 (median 0.56). The mean BRE of the consensus estimates after group discussion was 0.82 (median 0.50). The calculation of the statistics followed the same procedure as for the previous research question. A summary is presented in Table 7.

The analysis of a possible difference in accuracy between the statistical combination of individual estimates and the group consensus estimates yielded a *p*-value of 0.07 and an effect-size of 0.11.

5.3. RQ3: Are group consensus estimates more accurate than the existing individual estimation method?

The mean BRE of the tasks completed using the existing individual estimation method was 0.78 (median 0.33), compared to a mean BRE of 0.82 (median 0.50) for the planning poker tasks. For the statistical test of difference between the two groups, a Kruskal–Wallis test (Wonnacott and Wonnacott, 1990) was performed on the medians. A summary for both groups is found in Table 8.

The analysis of difference between the two study groups yielded a *p*-value of 0.77, and a size of effect of 0.03.

Regarding any difference in estimation bias, The BREbias values are presented in Fig. 1 (see also Table 5).

An interesting finding emerges. The median BREbias for both the planning poker group and the control group is 0.00, which indicates that the typical case is estimated perfectly on target.

However, the mean BREbias of the planning poker tasks was 0.33, compared to −0.04 for the control group (see also Table 5).

5.4. RQ4: Does the introduction of a group technique for estimation affect other aspects of the developers' work, when compared to the individual estimation method?

Thirty-four of the 53 tasks studied involved changing code. For seven of the 34 tasks that involved code changes, the developers did not tag the associated code repository checkins. Interviews revealed that this could sometimes happen for minor changes. Hence, the analysis below is based on 27 valid data points.

Table 5
Key results

	Initial estimate	Statistical combination (h)	Estimate (h)	Actual effort (h)	BRE statistical combination	BRE	BREbias
Planning poker (n = 24)							
Mean	6.6	6.3	7.1	10.4	0.94	0.82	0.33
Median	5.0	6.0	6.0	8.0	0.56	0.50	0.00
Control group (n = 29)							
Mean	5.3		5.3	6.1		0.78	−0.04
Median	4.0		4.0	4.0		0.33	0.00

Table 6
RQ1 results

N	24
Statistical combination (mean hours)	6.3
Group consensus (mean hours)	7.1
Difference	0.8
Pooled StDev	4.6
p-Value	0.04
Size of effect (*d*)	0.16

Table 7
RQ2 results

N	24
BRE statistical combination (mean)	0.94
BRE group consensus (mean)	0.82
Difference	0.12
Pooled StDev	1.02
p-Value	0.07
Size of effect (*d*)	0.11

Table 8
RQ4 results

BRE planning poker group (mean, $n = 24$)	0.82
BRE control group (mean, $n = 29$)	0.78
Difference	0.04
Pooled StDev	1.22
p-Value (Kruskal–Wallis)	0.77
Size of effect (*d*)	0.03

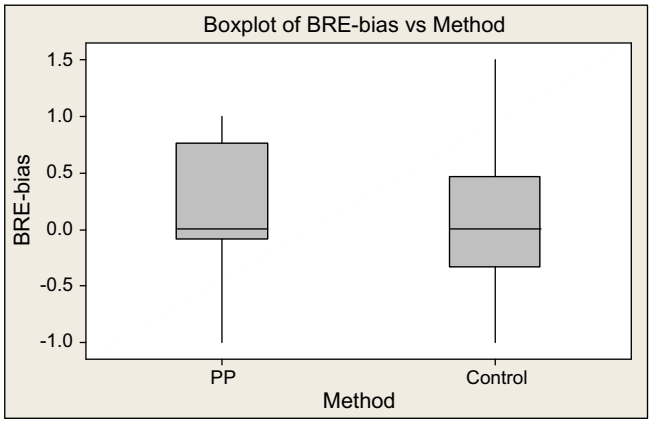

Fig. 1. BREbias.

Table 9
Mean values of key measures compared

Measure	Mean value control	Mean value PP
Estimate	7.5	7.6
Actual	8.1	12.8
ADD	246	326
CH	23.5	36.0
DEL	66.5	50.9
SIZE1	336	413
SIZE2	17.0	12.3
SIZE3	45.3	39.9
ACS	21.6	14.4
DCS	14.1	6.8
AOR	209	206
DOR	65	50
SZAFF	224	151

An overview of the results of the code analysis is presented in Table 9.

The mean actual effort for both groups is somewhat higher for this subset than for the complete set of tasks, which is presented in Table 5. This is not surprising, because tasks that involve changes to the code are usually larger than other tasks. The mean actual effort for the planning poker tasks involving code change is 12.8 h, compared to 10.4 h for the complete planning poker subset. Similarly, the mean actual effort of the control group is 8.1 h for the tasks involving code, compared with 6.1 h for all control tasks. Research question 4A asked: *Are there differences between the planning poker tasks and the control group tasks related to the size of the changes?*

The control group has a higher mean value for the SIZE3 measure than the planning poker group (see Table 9). Given this, it appears that the control group tasks took less time *and* were larger than the planning poker tasks. However, this difference in size is not statistically significant using a Kruskal–Wallis test ($p = 0.59$). The individual data points of SIZE3 vs. change effort are depicted in Fig. 2.

In order to further explore research question 4A, we fitted two regression models of the type discussed in Section 4.4, one that included isControl (the group indicator, refer to Table 3) only, and one that added the size variable, SIZE3. The results are summarized in Table 10.

As can be seen, when accounting for size (the second model), the difference in change effort between the groups become clearer (the *p*-value of the isControl variable decreases), and is statistically significant at the 0.1 level. In other words, there is initial evidence that, after controlling for size, effort expended on tasks estimated by planning poker is greater than effort expended on tasks in the control group.

Returning to the summary statistics in Table 9, we observe that planning poker tasks involved more changes (CH) and fewer deletions (DEL) to existing code. These factors can account for the observed difference in change effort.

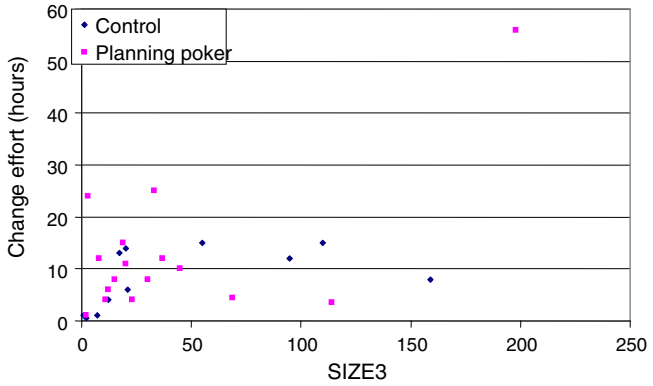

Fig. 2. Change effort and size.

Table 10
Models of change effort, without and with size measure included as covariate

Variable	Coefficient	*p*-Value	Deviance
Intercept	2.55	<0.0001	210
isControl	−0.449	0.25	
Intercept	2.12	<0.0001	139
SIZE3	0.00795	0.0004	
isControl	−0.482	0.10	

Table 11
Models for change complexity

Variable	Coefficient	*p*-Value	Deviance
Intercept	1.71	<0.0001	76
DEL	−0.0242	<0.0001	
CH	0.0193	<0.0001	
ACS	0.0184	0.016	
DOR	0.0166	0.0005	
Intercept	1.79	<0.0001	92
DEL	−0.017	0.0003	
CH	0.0198	<0.0001	
AOR	0.00136	0.0081	
DCS	0.0493	0.0018	

Research question 4B asked: *Are there differences between the planning poker tasks and the control group tasks related to the complexity of the changes*?

To analyze possible differences in complexity, we entered all complexity measures into the model presented in Section 4.4, and applied a variable selection method called backward elimination to attempt to identify factors that explain change effort variations. The results are presented in the top half of Table 11.

Measures of changed lines (CH), added control statements (ACS), and deleted out-of-class references (DOR) contribute positively to change effort. The measure of deleted lines (DEL) contributes negatively to change effort. When entering these measures of change complexity into the model, the estimation method (isControl) was no longer related significantly to change effort.

Thus, there is no evidence that effort expended on tasks estimated by planning poker is greater than effort expended on tasks in the control group after controlling for change complexity.

Note that the measures selected by the backward elimination procedure may have been influenced by correlations between the measures. ACS is correlated heavily with AOR and DOR is correlated heavily with DCS; hence, it is likely that AOR and DCS will provide almost as good explanatory power as ACS and DOR. We confirmed this by refitting a model forcing in the AOR and DCS variables; see the results in the bottom half of Table 11.

Given the above, we answer research question 4B positively: there are differences in change complexity between the planning poker tasks and the control tasks.

These differences in complexity might explain differences in effort. The underlying reasons for differences in complexity will be explored in the next section.

A final observation is that there is no evidence that the observed bias between the groups with respect to the mean size of affected modules resulted in differences in change effort.

5.5. Results from the participant interviews

The interviews provided information that will be used in the discussion section to try to explain the results presented above. The interviews focused on (a) background information regarding project priorities, (b) specific questions regarding planning poker, and (c) the participants' perception of differences between the planning poker technique and the individual estimation technique.

Table 12
Perceived importance of project parameters

Parameter	Mean	Median	StDev
Customer satisfaction	1.8	2.0	0.8
Functionality	2.0	2.0	0.6
Quality	2.2	2.0	0.8
Schedule	2.7	2.5	0.8
Effort	3.0	2.5	1.3

Table 13
Perceived influence when changing opinion

Parameter	Mean	Median	StDev
New information	1.8	2.0	0.4
Pressure from seniors	2.7	2.0	1.2
Desire for consensus	3.3	3.5	0.8

Table 14
Perceived differences of planning poker compared to control group tasks

Property	Rating (mean)	Median	StDev
Suitability for identifying task challenges	1.5	1.5	0.5
Suitability for identifying subtasks	1.7	1.5	0.8
Effort spent on estimation	1.8	2.0	0.4
Motivation to follow estimates	2.0	2.0	0.6
Estimation accuracy	2.2	2.0	0.8
Effort spent on analysis and design	2.8	2.5	1.0
Effort spent on refactoring of code	3.0	3.0	0.6
Effort spent on clarifying tasks during implementation (i.e. not including the estimation phase)	3.2	3.0	0.8

The participants were asked to rate how they perceived the priorities of the project used in this study on a five-point Likert-scale (1 = very important, 2 = important, 3 = of medium importance, 4 = somewhat important, 5 = not important). The respondents were free to use their personal interpretation of parameters such as quality and functionality. The results are displayed in Table 12.

There are several reasons for why people change their opinion about the estimate of a task after group discussion (Brown, 2000). Some of the more common reasons are (a) pressure (direct or perceived) from seniors, (b) new information revealed, or (c) a desire for consensus.

The participants were asked to rate how much these reasons affected their estimates on a five-point Likert-scale (1 = influence in all tasks, 2 = influence in most tasks, 3 = influence in about half of the tasks, 4 = influence in some tasks, 5 = influence in none of the tasks). The results are displayed in Table 13.

The participants were interviewed regarding possible differences induced by the planning poker technique when compared to the individual estimation method (control group). They were asked to rate their perception of whether, and if so how, several aspects of their work was influenced. This included both effort spent in various phases and suitability. They rated these aspects on a five-point Likert-scale (1 = much more, 2 = more, 3 = similar, 4 = less, 5 = much less). The results are summarized in Table 14.

6. Discussion

In general, small differences in estimation accuracy were found between the groups, whether the comparison was between a statistical combination of individual estimates and group consensus estimates for the planning poker tasks (choice shift), or between planning poker tasks and the control group.

Interestingly, there appeared to be a difference between the planning poker tasks and the control tasks that was related to change size and change complexity.

6.1. RQ1: Are group consensus estimates less optimistic than the statistical combination of individual expert estimates?

When we looked in isolation at the tasks estimated with planning poker, the results indicated a slight shift in choices that showed a reduction of optimism after group discussion. For these tasks, there

was an initial individual bias towards optimism, as in other studies (Buehler et al., 2005). However, in our study, this optimism was not increased by group discussion. Rather, we found the opposite, that optimism was reduced, as in a previous study on software estimation (Moløkken-Østvold and Jørgensen, 2004). However, note that the effect must be considered small (Cohen's $d < 0.20$).

The more common reasons for a change of opinion regarding the estimate of a task following group discussion have already been noted. The results of the interviews presented in Section 5.5 show that the respondents rated new information as the most important reason for changing their minds (mean response 1.8, and standard deviation of 0.4). However, pressure (2.7) and desire for consensus (3.3) were also rated as important for changes of opinion for about half of the tasks.

6.2. RQ2: Are group consensus estimates more accurate than the statistical combination of individual expert estimates?

There were indications of a slight shift towards increased accuracy when comparing consensus estimates with the statistical combination of individual estimates. This comes as a direct function of an initial bias towards optimism in the individual estimates. When, as found with respect to RQ1, this optimism was reduced, accuracy increased. However, the size of effect is considered small (Cohen's $d < 0.20$).

When exploring differences between a group's consensus estimate and the statistical combining of individual estimates, a relevant property is the initial (dis)agreement of the estimates for each task. This (dis)agreement can be measured by the standard deviation (StDev). For the 24 tasks estimated with planning poker, the StDev of the individual estimates varied from 0.00 to 6.65, and the median and mean StDev were 2.23 and 2.20, respectively.

However, initial agreement on the part of the estimators (reflected in a low standard deviation) did not entail more accurate group consensus estimates. A correlation test of standard deviation (StDev) of individual estimates, and the accuracy (BRE) of the group estimates resulted in a Pearson correlation of -0.54 and a p-value of 0.80.

6.3. RQ3: Are group consensus estimates more accurate than the existing individual estimation method?

The planning poker and control group had fairly similar estimation accuracy.

Regarding the observed difference in the *direction* of inaccuracy between the planning poker tasks and the control tasks, as seen by the difference in mean BREbias, one possible explanation is that some of the planning poker tasks were, purely by chance, more difficult to complete. Even though the tasks were assigned at random to the two study groups, anomalies may appear, especially in datasets of this size.

The mean of the *initial estimates*, i.e. estimates made before the assignment of the estimation procedure, was 5.3 h (median 4 h) for the control group, compared to 6.6 h (median 5 h) for the tasks estimated using planning poker. So, it is possible that the planning poker tasks were a bit more difficult, because they had initial estimates that were somewhat larger.

Comparison of the actual effort of the tasks in the two study groups yields an interesting observation. The average size in the control group tasks was 6.1 h (median 4 h), compared to 10.4 h (median 8 h) in the planning poker tasks. While the median *initial estimates* were 25% larger in the planning poker group, the difference in median *actual effort* was 100%. Even though it is possible that initially, the tasks in the planning poker tasks carried a somewhat larger workload, this cannot account for the observed difference in actual effort between the groups.

The observations of differences in actual effort between planning poker and control tasks was unexpected, and, as stated in Sections 1 and 2, there is very little research on the combining of estimates for comparison. At this stage, we can only speculate about a set of interacting causes that may explain our observations:

6.3.1. Group discussion identifies subtasks and complexity

Previous studies have shown that groups are able to identify more tasks than individuals (Moløkken-Østvold and Jørgensen, 2004). When the group discusses a task (as when estimating using planning poker), they are likely to look at it from different angles, especially if they have diverse backgrounds (Surowiecki, 2004). They may offer different perspectives on a task and identify different subproblems.

Software engineering textbooks (Boehm, 1981; Kitchenham, 1996) and papers (Fairley, 2002; Boehm, 1984; Hughes, 1996) frequently mention forgotten tasks as major obstacles to successful estimation by experts. Several estimators who discuss the same task will identify at least as many subtasks as any single estimator alone. It might be that this happened for the tasks that were estimated using planning poker.

As seen from the results of the interviews presented in Section 5.5, the participants perceived that planning poker influenced their work most with respect to identifying subtasks and challenges. They also thought that they spent more time estimating and that they were more motivated to match their estimates. Even though estimation accuracy did not increase when using planning poker, the participants believed that it did. The three areas in which the participants did not perceive any changes were related to effort spent on analysis and design, refactoring, and clarifications.

The code analysis revealed that more effort was spent on performing complex changes in the planning poker tasks. This might have been induced by the group discussion.

6.3.2. Anchor-effect from individual estimates

Even if the participants identified more subtasks and complexity during group discussion, it is possible that they were not able to make sufficient adjustments on the basis of this new information when seeking consensus estimates. Even though the group decision exhibited decreased optimism when compared to the statistical combining of individual estimates, this was, for some tasks, not sufficient, and they were underestimated.

It is probable that the initial individual estimates acted as anchors (Jørgensen and Sjøberg, 2004) when group consensus was sought. Participants were frequently willing to increase their estimates somewhat, e.g. by about one hour, but it seldom happened that consensus estimates deviated substantially from the statistical combination. As seen from results of the interviews, presented in Section 5.5, it was important for the participants to reach a consensus.

6.3.3. Priority of scope over effort and schedule

If we assume that more task work and greater complexity was identified during planning poker discussions (as follows from explanation 1), and the participants have their original estimates (and group mean) as anchors (as follows from explanation 2), this may lead to underestimated tasks.

When a task is estimated in a group (as with planning poker), and then handed to an individual, that individual must address all the aspects of the work discussed by the group when implementing the solution. By contrast, individual programmers who estimate and plan alone (as in the control group) have a more limited range of work aspects to address.

When underestimated tasks are encountered, the implementation will be affected, according to how priority is assigned to scope,

effort, schedule, etc. A recent study found that software professionals gave priority to "project scope" when defining project success (Agarwal and Rathod, 2006). The professionals in that study, independent of their role in the company, stated that scope was more important than cost (effort) or time (schedule) when asked to state their priorities. We have also recently conducted a study in Norway, where it was found that (lack of) estimation accuracy did not affect perceived project success (Furulund, 2007).

As seen from the results of the interviews presented in Section 5.5, effort and schedule were perceived as least important by the participants, while functionality and customer satisfaction were perceived as most important.

Thus, if more work is identified during discussion, programmers may feel inclined to expend more effort in order to implement it. It might just be that for some of the planning poker tasks, work such as the restructuring of code uncovered in our analysis caused some overruns.

6.4. RQ4: Does the introduction of a group technique for estimation affect other aspects of the developers' work, when compared to the individual estimation method?

Differences in effort between groups were amplified when controlling for *change size*: More effort was expended on the planning poker tasks *and* the sizes of these tasks were smaller than in the control group.

Differences with respect to the *complexity* of the tasks can explain the difference. Changes made in the tasks estimated by planning poker were more complex, as manifested in measures of the changed code.

This observation must be seen together with the respondents' claim that planning poker was more suitable for identifying subtasks and challenges. It is possible that the planning poker method itself influenced the way the developers translated the change requests into working code.

6.5. Study validity

In their framework for analysing the accuracy of software estimation (Grimstad and Jørgensen, 2006), Grimstad and Jørgensen describe several factors that can have a major impact on the measured estimation error. Their top-level categories are: (1) estimation ability factors, (2) estimation complexity factors, and (3) measurement process factors.

When discussing the internal validity of our comparison of the planning poker and control groups (RQ3 and RQ4), many of the factors in the framework do not cause concern, because they are similar for both groups. Examples of these are (a) the project manager's ability to control costs, (b) client and subcontractor performance, (c) completeness and certainty of the information upon which the estimates were based, (d) project priorities, (e) project member skill, (f) inherent complexity of project execution, (g) experience with similar tasks, (h) experience of the system under consideration, (i) flexibility in product and process execution, (j) terminology and measures, and (k) the recording of data.

These factors were similar in both groups and did not have any effect, because all tasks in the study were taken from the same project, with, e.g., the same client, participants, and prioritizations.

In addition, the *isolation strategy* used was randomization, which is the most powerful strategy. This approach addresses concerns such *as skill in the selection of estimation approach*, because this was assigned randomly. However, as described previously, even randomization is no guarantee that the samples will have similar properties with respect to all factors. As seen, the *sizes* of the initial estimates of the tasks were not entirely similar in the groups, even though the variations were small.

Perhaps the most challenging issue concerns one of the estimation ability factors; namely, *skill in the use of estimation approach* (Grimstad and Jørgensen, 2006). Since the estimators were more familiar with their existing individual estimation method (control group), it might be that their skill in employing this was superior to their skill in using planning poker. However, we do not believe that this factor had any major impact, because planning poker is a straightforward and easy-to-use approach that should not require a steep learning curve. We performed an analysis of the estimation accuracy of the planning poker tasks to determine whether there was any learning. It was found that the estimation accuracy was similar for the planning poker tasks throughout the entire study.

The internal validity of the choice shift (RQ1 and RQ2) research questions is relatively unproblematic, because it involved several estimates for the same task.

Regarding external validity, only one project team was studied. Therefore, several factors must be considered when generalizing. In particular, factors such as team motivation (Brown, 2000) and team composition (Surowiecki, 2004) will probably have a large impact on the results. For example, a team that lacks diversity and motivation may increase an optimistic bias instead of reducing it. Most important perhaps, is that this study was on task estimation, which has properties other than user story estimation (for which planning poker is recommended), project estimation, and factors related to bidding.

Finally, regarding generalization, it is important to note that the tasks studied here were relatively small and were to be performed in an agile project environment. At this time, we have no opportunity to assess the merits of using planning poker in other project environments.

A general concern is that the study had a relative small sample size with respect to statistical analysis. The source code data, especially, contained few data points with large variances; hence the analysis is sensitive to the values of small groups of data points. In particular, one data point has a large influence on the mean change effort of planning poker tasks. However, removing this data point does not change the observation that change effort is greater, by median or mean value, for planning poker tasks. In addition, when controlling for size, this data point can no longer be considered an outlier. We therefore included the data point in our analysis.

Given the above reservations, this study must be interpreted with care and used primarily in combination with previous studies on group estimation (presented in Section 2) as a stepping stone for further research.

7. Conclusions

Previous reviews of the literature and experiments have concluded that it does not seem to be very important which of a set of structured methods for combining estimates is used in order to achieve accuracy (Fischer, 1981). The Delphi technique is probably the best studied example, and though it has been found to outperform unstructured groups, there is no evidence that it outperforms other structured techniques (Rowe and Wright, 1999). There are also general findings to the effect that group performance is increased when motivation exists (Brown, 2000) and that group goals can increase productivity (Buehler et al., 2005).

On the surface, planning poker has several properties that, in theory, should make it suitable for estimation; for example, the possibility of combining knowledge from diverse sources (Surowiecki, 2004), the use of iterative techniques, and the fact that estimates are revealed simultaneously in order to reduce the impact of social comparison (Brown, 2000).

Considering a summary of our findings and combining them with previous studies, we may conclude tentatively that planning poker reduces optimism when compared to the statistical combining of individual estimates and is also, in some cases, more accurate than the unstructured combining of estimates in a group.

In this study, the set of control tasks in the same project were estimated by individual experts with accuracy similar to that of the estimates of the tasks when using planning poker. Moreover, for both the planning poker and control groups, the median estimation bias indicated that both groups had fairly unbiased estimates. In addition, as seen in our study, group discussion (facilitated by planning poker) may have certain positive side effects that, at this stage, we cannot fully explain. An interesting issue, derived from the analysis of code, is whether the use of planning poker leads to an increased focus on the quality of the code.

Equally important as findings from the quantitative data, the project team seemed to receive the planning poker technique very well. They found that the technique was useful for discussing implementation strategies for each task and that it provided a better overview of what each developer was working on. Given that it is difficult to measure the full effect of the knowledge sharing aspect of planning poker, we cannot provide any empirical results on whether the benefit exceeds the work and effort it takes to conduct this technique compared to individual estimating. However, the team decided to implement the planning poker technique for all forthcoming tasks in the project.

Future studies might seek to complement these findings by investigating projects with different constraints regarding team size and client, and using planning poker for estimating user stories.

In addition, we should investigate how planning poker can be combined with complementary techniques for tracking time/cost, i.e., having developers report progress daily (at the stand-up meeting), having all tasks posted on the wall to visualize the total work load for the sprint, and/or using a burn-down chart to visualize progress.

It is also important to compare planning poker with more structured techniques, such as Delphi, and to investigate whether planning poker affects other technical or social aspects, such as the quality of the code or the accountability of the team.

Acknowledgements

We thank all the subjects and the management of the studied company for providing data, and Magne Jørgensen, Stein Grimstad, Mike Cohn, Amund Tveit, Kristian Marius Furulund, and Chris Wright for valuable comments. This research was funded by the Research Council of Norway under the project INCO.

References

Agarwal, N., Rathod, U., 2006. Defining "success" for software projects: an exploratory revelation. International Journal of Project Management 24, 358–370.

Arisholm, E., 2006. Empirical assessment of the impact of structural properties on the changeability of object-oriented software. Information and Software Technology.

Armstrong, J.S., 2006. How to make better forecasts and decisions: avoid face-to-face meetings. The International Journal of Applied Forecasting Fall, 3–15.

Aronson, E., Wilson, T.D., Akert, R.M., 1999. Social Psychology, third ed. Addison-Wesley Educational Publishers Inc.

Atkinson, R.L., Atkinson, R.C., Smith, E.E., Bem, D.J., Nolen-Hoeksema, S., 1996. Hilgard's Introduction to Psychology, 12th ed. Harcourt Brace College Publishers, Orlando.

Berndt, D.J., Jones, J.L., Finch, D., 2006. Milestone markets: software cost estimation through market trading. In: 39th Hawaii International Conference on System Science, Hawaii.

Boehm, B., 1981. Software Engineering Economics. Prentice Hall PTR.

Boehm, B., 1984. Software engineering economics. IEEE Transactions on Software Engineering 10, 4–21.

Briand, L.C., Daly, J.W., Wust, J.K., 1999. A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on Software Engineering 25, 91–121.

Brown, R., 2000. Group Processes, second ed. Blackwell Publishers.

Buehler, R., Messervey, D., Griffin, D., 2005. Collaborative planning and prediction: does group discussion affect optimistic biases in time estimation? Organizational Behaviour and Human Decision Processes 97, 47–63.

Cohen, J., 1969. Statistical Power Analysis for the Behavioral Sciences. Academic Press, Inc., New York.

Cohn, M., 2005. Agile Estimating and Planning. Addison-Wesley.

Fairley, D., 2002. Making accurate estimates. IEEE Software 19, 61–63.

Fenton, N.E., 1995. Software Metrics. Thompson Computer Press, London.

Fischer, G.W., 1981. When oracles fail—a comparison of four procedures for aggregating subjective probability forecasts. Organizational Behaviour and Human Performance 28 (August), 96–110.

Fluri, B., Gall, H., 2006. Classifying change types for qualifying change couplings. In: Proceedings of 14th IEEE International Conference on Program Comprehension, Athens, Greece, June, pp. 14–16.

Furulund, M.K., 2007. Empirical Research on Software Effort Estimation Accuracy, Master Thesis, Department of Informatics, University of Oslo.

Graves, T.L., Mockus, A., 1998. Inferring change effort from configuration management databases. In: Proceedings of the 5th International Symposium on Software Metrics, pp. 267–272.

Grenning, J., 2002. Planning Poker or How to avoid analysis paralysis while release planning.

Grimstad, S., Jørgensen, M., 2006. A framework for the analysis of software cost. In: ISESE 2006, Rio de Janeiro, Brazil, pp. 58–65.

Hanson, R., 1999. Decision markets. IEEE Intelligent Systems 14, 16.

Haugen, N.C., 2006. An empirical study of using planning poker for user story estimation. In: Agile 2006 Conference (Agile'06).

Helmer, O., 1966. Social Technology. Basic Books, New York.

Höst, M., Wohlin, C., 1998. An experimental study of individual subjective effort estimations and combinations of the estimates. In: 20th International Conference on Software Engineering, Kyoto, Japan, pp. 332–339.

Hughes, R.T., 1996. Expert judgment as an estimating method. Information and Software Technology, 67–75.

Humphrey, W.S., 1990. Managing the Software Process. Addison-Wesley Publishing Company, Inc.

Hunt, J.W., McIlroy, M.D., 1976. An algorithm for differential file comparison. Computing Science Technical Report, vol. 41, Bell Laboratories.

Jørgensen, M., 1995. Experience with the accuracy of software maintenance task effort prediction models. IEEE Transactions on Software Engineering 21, 674–681.

Jørgensen, M., 2005. Practical guidelines for expert-judgment-based software effort estimation. IEEE Software 22, 57.

Jørgensen, M., Sjøberg, D.I.K., 2004. The impact of customer expectation on software development effort estimates. International Journal of Project Management 22, 317–325.

Kitchenham, B., 1996. Software Metrics: Measurement for Software Process Improvement. NCC Blackwell, Oxford.

Liden, R.C., Wayne, S.J., Sparrowe, R.T., Kraimer, M.L., Judge, T.A., Franz, T.M., 1999. Management of poor performance: a comparison of manager, group member, and group disciplinary decisions. Journal of Applied Psychology 84, 835–850.

MacKenzie, D., Eggert, P., Stallman, R., 2003. Comparing and Merging Files with Gnu Diff and Patch. Network Theory Ltd.

McCabe, T., 1976. A complexity measure. IEEE Transactions on Software Engineering SE-2, 308–320.

Miyazaki, Y., Takanou, A., Nozaki, H., Nakagawa, N., Okada, K., 1991. Method to estimate parameter values in software prediction models. Information and Software Technology 33, 239–243.

Moløkken-Østvold, K., Jørgensen, M., 2004. Group processes in software effort estimation. Empirical Software Engineering 9, 315–334.

Myers, R.H., Montgomery, D.C., Vining, G.G., 2002. Generalized Linear Models with Applications in Engineering and the Sciences, Wiley Series in Probability and Statistics.

Niessink, F., van Vliet, H., 1998. Two case studies in measuring software maintenance effort. In: International Conference on Software Maintenance, pp. 76–85.

Passing, U., Shepperd, M., 2003. An experiment on software project size and effort estimation. In: International Symposium on Empirical Software Engineering (ISESE 2003), Frascati – Monte Porzio Catone (RM), Italy, pp. 120–129.

Powell, C., 2003. The Delphi technique: myths and realities. Journal of Advanced Nursing 41, 376–382.

Rowe, G., Wright, G., 1999. The Delphi technique as a forecasting tool: issues and analysis. International Journal of Forecasting, 353–375.

Rowe, G., Wright, G., 2001. Expert opinions in forecasting: the role of the Delphi technique. In: Armstrong, J.S. (Ed.), Principles of forecasting. Kluwer Academic Publishers, Boston.

Surowiecki, J., 2004. The Wisdom of Crowds. Doubleday.

Van de Ven, A.H., Delbecq, A.L., 1971. Nominal versus interacting group processes for committee decision making effectiveness. Academy of Management Journal 14, 203–212.

Wonnacott, T.H., Wonnacott, R.J., 1990. Introductory Statistics, fifth ed. John Wiley & Sons, Inc.

Zuber, J.A., Crott, H.W., Werner, J., 1992. Choice shift and group polarization: an analysis of the status of arguments and social decision schemes. Journal of Personality and Social Psychology 62, 50–61.

Kjetil Moløkken-Østvold is a Senior Partner at Conceptos IT Development in Norway (www.conceptos.no). Areas of expertice include project management, cost estimation, process improvement and agile development. He has experience as Assistant Director and Postdoctoral researcher at Simula Research Laboratory, and as founder and manager of Project Economics. Moløkken-Østvold received his Masters degree in Informatics from the University of Oslo in June 2002. In December 2004 he received the PhD-degree in Informatics from the University of Oslo and Simula Research Laboratory.

Nils Christian Haugen received the MSc degree in computer science from the Norwegian University of Science and Technology in 1998. He has since worked as a consultant on software development and process improvement for companies in various industries. His main research interests are studies on analysis, estimating and planning in agile software development.

Hans Christian Benestad received the MSc degree in computer science from the Norwegian University of Science and Technology. He has 11 years industry experience on software development, process improvement, software tool design, and management. He is currently a PhD candidate at Simula Research Laboratory. His main research interest is empirical studies on evolution of software systems.