The Impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments

Magne Jørgensen and Tanja M. Gruschke

Abstract—Inaccurate estimates of software development effort is a frequently reported cause of IT-project failures. We report results from a study that investigated the effect of introducing lessons-learned sessions on estimation accuracy and the assessment of uncertainty. Twenty software professionals were randomly allocated to a Learning group or a Control group and instructed to estimate and complete the same five development tasks. Those in the Learning group but not those in the Control group were instructed to spend at least 30 minutes on identifying, analyzing, and summarizing their effort estimation and uncertainty assessment experience after completing each task. We found that the estimation accuracy and the realism of the uncertainty assessment were not better in the Learning group than in the Control group. A follow-up study with 83 software professionals was completed to better understand this lack of improvement from lessons-learned sessions. The follow-up study found that receiving feedback about other software professionals' estimation performance led to more realistic uncertainty assessments than receiving the same feedback of one's own estimates. Lessons-learned sessions, not only in estimation contexts, have to be carefully designed to avoid wasting resources on learning processes that stimulate rather than reduce learning biases.

Index Terms—Cost estimation, process implementation and change, review and evaluation, software psychology.

1 Motivation

A recent questionnaire-based survey of more than 1,000 IT professionals [1] reports that two out of the three most important causes of IT-project failure are perceived to be related to poor effort estimation. This perception is supported by empirical studies of software development effort overruns and overconfidence. The typical effort overrun seems to be about 30 percent [2] and 90 percent confidence (or "almost sure") that the actual effort will be included in a minimum-maximum effort interval typically corresponding to a 60-70 percent interval hit rate [3]. Overconfidence in the accuracy of effort estimates typically means that the project plan is not reflecting the underlying uncertainties related to the use of effort. Clearly, both software vendors and clients would benefit from more accurate effort estimates and more realistic assessments of the uncertainty of these estimates. One possible way of achieving higher accuracy and more realistic uncertainty assessments is to develop and adopt better learning processes. One candidate for improved learning is the use of individual lessons-learned sessions, i.e., structured reviews of one's own estimation experience. The learning effect of such lessons-learned sessions is the topic of this paper.

Judging by the number of scientific studies, it seems that software engineering researchers believe that estimation learning processes should focus on improving formal estimation models. This has been the main approach to learning and improvement among software researchers since at least the 1960s [4]. This strong focus on the improvement of formal estimation models can hardly be defended by its success. Even the use of unstructured and unsupported judgment-based effort estimation ("expert estimation") seems, on average, to yield effort estimates that are just as accurate as those generated by the use of sophisticated formal estimation models (see our review in [5]). In addition, formal effort estimation models are not much used by the software industry [6]. The use of the Personal Software Process (PSP) [7] provides, in our opinion, an illustration of the limitations of an estimation learning process that is based mainly on the improvement of formal effort estimation models. Independent studies with control groups report a surprising lack of improvement in estimation accuracy when applying the PSP process [8], [9]. Note that, irrespective of these results, we believe that the PSP is based on many sound learning principles and may be useful for several purposes. The PSP and the lessons-learned sessions that we introduce in our study share many of the same learning principles, e.g., the tracking and feedback of estimates and actual effort, the tracking of estimation problems, and emphasis on reflections on how to avoid the problems in the future.

There has been much work on learning from experience in software organizations, e.g., work on experience databases [10], [11], [12], [13] and work on project experience reviews [14], [15]. However, none of these studies report results on how the use of different types of lessons-learned processes improves judgment-based effort estimation and few of them compare the learning effect with on-the-job learning. The comparison with on-the-job learning is, we believe, essential if we are to avoid attributing to the introduced change in learning processes learning that would have happened without it. Therefore, in our study,

- *M. Jørgensen is with the Simula Research Laboratory and the University of Oslo, Statsråd Ihlensv. 14A, 2010 Strømmen, Norway. E-mail: magnej@simula.no.*
- *T.M. Gruschke is with KnowIT Objectnet, Ruseløkkveien 14, 0251 Oslo, Norway. E-mail: tanjagru@gmail.com.*

Manuscript received 30 Apr. 2008; revised 10 Nov. 2008; accepted 10 Dec. 2008; published online 9 Jan. 2009.
Recommended for acceptance by A. Mockus.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2008-04-0164.
Digital Object Identifier no. 10.1109/TSE.2009.2.

0098-5589/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

we decided to compare the improvement in learning that followed lessons-learned sessions with that of a control group that represented on-the-job learning.

The study reported in [16] compares the accuracy of the estimation of software development effort of those that receive explicit feedback that compares the estimated and the actual effort (feedback on outcome), with that of a nonfeedback group. The results from that study suggest that estimators that received feedback on outcome produced estimates that were no more accurate than those that did not receive such feedback. Studies on human judgment from other domains support this disappointing lack of improvement in estimation accuracy as a result of feedback on outcome; see, for example, [17], [18], [19]. The unsatisfactory effect of feedback on outcome motivated us to emphasize lessons-learned sessions, i.e., a process that includes more analysis than examination of the outcome feedback alone. The structure we imposed on these lessons-learned sessions (see Section 2) is based on the finding that information on how different events and variables are related to effort overruns are reported to be required for improved judgment accuracy in several domains [17], [18], [19], [20].

Lessons-learned sessions should be designed to solve problems pertaining to learning from experience that are typically part of on-the-job learning, such as those reported in [21]:

- High time pressure. The next task should be started as soon as possible.
- Low reflection on mistakes.
- Lack of objective feedback.
- Lack of knowledge about how to summarize lessons learned.
- Perception of the task as unique and not relevant to learn from.

The lessons-learned sessions we introduce in Section 2 address at least three of these problems. They remove the high time pressure, force the software professionals to reflect on mistakes, and provide some relevant objective feedback. It is, however, possible that the two remaining problems, i.e., the lack of knowledge about how to summarize lessons learned and the perception of the task as unique and not relevant to learn from, are not addressed sufficiently. If these two problems are essential to achieving positive effect from lessons-learned sessions, if there is insufficient objective feedback, or if there are other learning problems that are not addressed by lessons-learned sessions, we may observe no or even negative learning from our lessons-learned sessions.

The remainder of this paper is organized as follows: Section 2 describes the design of the study, including the format of the lessons-learned sessions. Section 3 describes the results of the study. Section 4 discusses limitations of the study and tries to explain the findings. That section includes a description of a follow-up study that was intended to shed more light on the findings. Section 5 concludes.

2 Design

2.1 Main Research Question

The main research question addressed in this study is the following:

> *To what extent do lessons-learned sessions (of the type implemented in this study) lead to improved accuracy of effort estimation and improved realism of uncertainty assessment in comparison to on-the-job learning?*

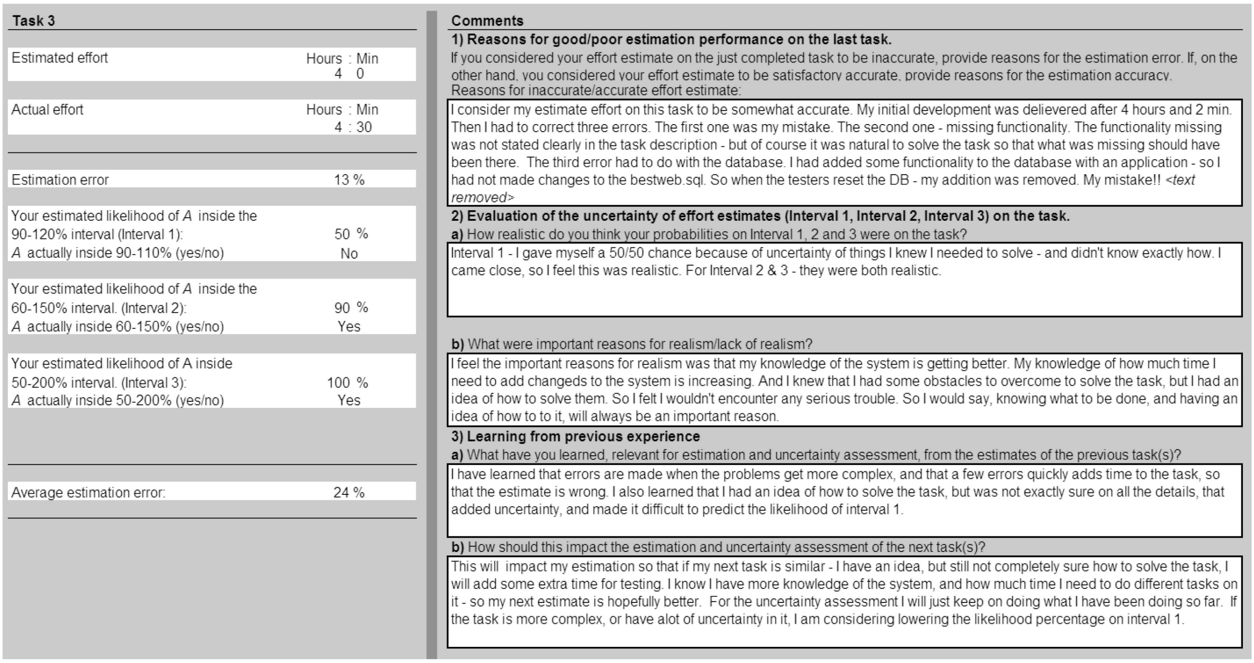
2.2 Structure of Lessons-Learned Sessions

The lessons-learned sessions evaluated in this study contain the following elements, provided immediately after the completion of each development task:

- Feedback on the estimated effort, the actual effort, and the estimation error of the task just completed.
- Feedback on the realism of the uncertainty assessments.
- Feedback on the mean estimation error of all preceding tasks.
- A required 30-minute learning session in which the following information should be provided by the software professional:
  - Perceived reasons for good/poor estimation performance.
  - Perceived realism of confidence in estimation accuracy.
  - Perceived reasons for realism/lack of realism of confidence in estimation accuracy.
  - Suggested lessons learned that are relevant for estimation and uncertainty assessment.
  - Suggestions on how the lessons learned should affect the estimation and uncertainty assessment of future task(s).

The feedback about the estimation performance and the format of the lessons-learned session is illustrated in Fig. 1. The fields on the left side of the vertical bar in Fig. 1 contain the feedback collected by the study-responsible researcher and given to the developer, i.e., the feedback about the estimated effort, the actual effort, the estimation error, the assessed likelihood of including the actual effort in different interval, the actual inclusion in the same intervals, and the mean estimation error of all previously completed tasks. No other feedback or performance information was given, but the developers themselves were, of course, free to collect all types of information. The fields on the right side of the vertical bar in Fig. 1 contain the analysis and lessons learned as provided by the developer. The questions on the right side, to be completed in sequence, provide the structure of the lessons-learned session. The feedback and lessons-learned-related information of all previous tasks was included in the same spreadsheet and easily accessible.

Notice that our lessons-learned session raises the same type of questions as in common industrial project review meetings. The postmortem reviews at Microsoft, for example, raise the questions: “What worked well in the last project, what went wrong, and what should the group do to improve in the next project?” [22]. An essential difference from many other types of project reviews is that we focus solely on individual learning in our study, while



Task 3

Estimated effort	Hours : Min 4 0
Actual effort	Hours : Min 4 : 30
Estimation error	13 %
Your estimated likelihood of *A* inside the 90-120% interval (Interval 1):	50 %
A actually inside 90-110% (yes/no)	No
Your estimated likelihood of *A* inside the 60-150% interval. (Interval 2):	90 %
A actually inside 60-150% (yes/no)	Yes
Your estimated likelihood of A inside 50-200% interval. (Interval 3):	100 %
A actually inside 50-200% (yes/no)	Yes
Average estimation error:	24 %

Comments

1) Reasons for good/poor estimation performance on the last task.

If you considered your effort estimate on the just completed task to be inaccurate, provide reasons for the estimation error. If, on the other hand, you considered your effort estimate to be satisfactory accurate, provide reasons for the estimation accuracy.

Reasons for inaccurate/accurate effort estimate:

I consider my estimate effort on this task to be somewhat accurate. My initial development was delievered after 4 hours and 2 min. Then I had to correct three errors. The first one was my mistake. The second one - missing functionality. The functionality missing was not stated clearly in the task description - but of course it was natural to solve the task so that what was missing should have been there. The third error had to do with the database. I had added some functionality to the database with an application - so I had not made changes to the bestweb.sql. So when the testers reset the DB - my addition was removed. My mistake!! *<text removed>*

2) Evaluation of the uncertainty of effort estimates (Interval 1, Interval 2, Interval 3) on the task.

a) How realistic do you think your probabilities on Interval 1, 2 and 3 were on the task?

Interval 1 - I gave myself a 50/50 chance because of uncertainty of things I knew I needed to solve - and didn't know exactly how. I came close, so I feel this was realistic. For Interval 2 & 3 - they were both realistic.

b) What were important reasons for realism/lack of realism?

I feel the important reasons for realism was that my knowledge of the system is getting better. My knowledge of how much time I need to add changeds to the system is increasing. And I knew that I had some obstacles to overcome to solve the task, but I had an idea of how to solve them. So I felt I wouldn't encounter any serious trouble. So I would say, knowing what to be done, and having an idea of how to to it, will always be an important reason.

3) Learning from previous experience

a) What have you learned, relevant for estimation and uncertainty assessment, from the estimates of the previous task(s)?

I have learned that errors are made when the problems get more complex, and that a few errors quickly adds time to the task, so that the estimate is wrong. I also learned that I had an idea of how to solve the task, but was not exactly sure on all the details, that added uncertainty, and made it difficult to predict the likelihood of interval 1.

b) How should this impact the estimation and uncertainty assessment of the next task(s)?

This will impact my estimation so that if my next task is similar - I have an idea, but still not completely sure how to solve the task, I will add some extra time for testing. I know I have more knowledge of the system, and how much time I need to do different tasks on it - so my next estimate is hopefully better. For the uncertainty assessment I will just keep on doing what I have been doing so far. If the task is more complex, or have alot of uncertainty in it, I am considering lowering the likelihood percentage on interval 1.

Fig. 1. The feedback and the lessons-learned session format.

most other project reviews focus on individual learning, group-based learning, and experience sharing. Our focus on individual learning is similar to that of the PSP.

It can be argued that reflecting on and learning from one's own work is the basis of many learning processes. If individual learning is not present, there is frequently not much experience of value to share either. However, the focus on individual learning in this study also means that the potential improvement in estimation accuracy that may result from group-based learning and experience sharing in lessons-learned sessions is not evaluated.

2.3 Measures

The measures used to evaluate estimation accuracy are the following:

$$Magnitude\ of\ Relative\ Error = MRE = |actual\ effort - estimated\ effort|/actual\ effort$$

$$Relative\ Error = RE = (actual\ effort - estimated\ effort)/actual\ effort.$$

The median MRE is our main measure of estimation accuracy. The lower the median MRE, the better is the estimation accuracy. We use the median RE as our main measure of estimation bias. A positive median RE indicates a tendency toward overoptimism, a negative median RE a tendency toward overpessimism, and an RE close to zero unbiased estimates. From previous experience, we expected that there would be a few MRE and RE-values (outliers) with a strong impact on the mean MRE and mean RE, i.e., that the mean MRE and RE would be misleading as the central value of a set of observations. That motivates our use of the more outlier-robust median MRE and RE.

The MRE and RE have their limitations and a lot of alternative accuracy measures have been proposed. For an overview and discussion, see [23]. We tested different variants of these accuracy measures on the data collected in this study, e.g., more symmetric accuracy measures, but found that they did not change the main results. We, therefore, decided to use the measures that are best known by the software engineering community, i.e., MRE and RE.

We measure the assessed uncertainty of an effort estimate by means of an effort prediction interval. An effort prediction interval is the combination of a stated confidence level (CL) and an effort minimum-maximum interval (EI). For example, a software developer may estimate that the most likely effort of a development task is 300 work-hours and it is 80 percent probable that the actual effort will be between 50 percent (150 work-hours) and 200 percent (600 work-hours) of the estimated most likely effort. Then, the minimum-maximum interval [150; 600] work-hours is the 80 percent confidence, prediction interval of the developer's effort estimate of 300 work-hours. In order to ease the analysis in our study, we standardized on the following three effort minimum-maximum intervals:

$$EI\text{-}1 = [90\% * estimated\ effort; 110\% * estimated\ effort]$$

$$EI\text{-}2 = [60\% * estimated\ effort; 150\% * estimated\ effort]$$

$$EI\text{-}3 = [50\% * estimated\ effort; 200\% * estimated\ effort].$$

These three effort minimum-maximum intervals were selected to reflect what we believed were typical narrow, medium-wide, and wide minimum-maximum effort intervals. For each task, we asked the developers to assess how confident he was, i.e., how probable he thought it was that the actual effort would fall inside each of these three minimum-maximum intervals. If, as in the previous example, a software developer estimated that the most likely effort was 300 work-hours, he would be asked to provide the (subjective) probabilities that the actual effort would fall inside the intervals [270; 330] work-hours (E1-1), [180; 450] work-hours (EI-2), and [150; 600] work-hours (EI-3). The developer may, for example, believe that it is 50 percent probable that the actual effort will be inside EI-1, 70 percent probable that it will be inside EI-2, and 95 percent probable that it will be inside EI-3. The CL of EI-1 is consequently 50 percent, the CL of EL-2 is 70 percent, and the CL of EL-3 is 95 percent.

The frequency of including the actual effort in an effort minimum-maximum interval is termed the "hit rate." In the long run, i.e., when estimating and assessing the uncertainty of many tasks, a developer who has realistic uncertainty assessments will have a mean confidence level that is similar to his hit rate. If his mean confidence level is higher than his hit rate, we describe the developer's uncertainty assessments as overconfident. If it is lower than the hit rate, we describe the developer's uncertainty assessments as underconfident. In earlier studies, e.g., [3], we have documented a strong tendency toward overconfidence in the accuracy of the effort estimates.

We measure the level of overconfidence of an effort minimum-maximum interval, e.g., EI-1, and a set of tasks as follows:

$$OverConfidence = \\ (mean\ confidence\ level - "hit\ rate")/"hit\ rate".$$

An OverConfidence value close to zero indicates that the uncertainty assessments, on average, are realistic, a negative value that they are underconfident, and a positive value that they are overconfident. Assume, for example, that a developer has provided the confidence levels (subjective probabilities) 80, 60, 90, 90, and 80 percent related to the E1-1 effort interval, for the estimates of the tasks $\mathrm{T1}\ldots\mathrm{T5}$, respectively. The mean confidence level of the developer's estimates related to E1-1 and $\mathrm{T1}\ldots\mathrm{T5}$ is then (80 percent + 60 percent + 90 percent + 90 percent + 80 percent)/5 = 80 percent. If the developer's uncertainty assessments are realistic, we would expect a hit rate of 80 percent, i.e., that four out of the five estimates lay inside the E1-1 minimum-maximum effort intervals. If the EI-1 minimum-maximum effort intervals, for example, included only one out of the five actual effort values, i.e., if we observed a hit rate of only 20 percent, his level of overconfidence would be (80 percent − 20 percent)/20 percent = 3.0 and would suggest a high level of overconfidence in the accuracy of his estimates. Notice that our measure of overconfidence may not work well as an indicator of overconfidence for as small a set of tasks as in the above example. Consequently, we will only use this measure as an indicator of differences in levels of confidence between groups that include the uncertainty assessments of larger sets of tasks.

The defined measure of overconfidence indicates only the ability of developers to assess the mean level of estimation uncertainty of a set of tasks. Also, relevant is the ability to distinguish between high and low uncertainty effort estimates in a set of tasks. For example, it is possible that the developers are strongly overconfident about the uncertainty of their effort estimates but are nevertheless able to distinguish between high and low uncertainty effort estimates. To assess the ability to distinguish between high and low uncertainty effort estimates, we apply the rank order correlation (Spearman rank order correlation) between confidence level and the estimation accuracy (MRE) for a minimum-maximum interval, e.g., EI-1, and a set of tasks:

$$CorrConfAcc = rank\ order\ correlation\ between\ confidence \\ level\ and\ MRE.$$

This measure uses the level of confidence as the uncertainty of an estimate as perceived by the software professional and the MRE-value as a substitute of the actual uncertainty of the effort estimate, i.e., the measure is based on the reasonable assumption that the higher the actual uncertainty of an effort estimate, the higher the typical estimation error will be. If a developer is able to rank his estimates in relation to the degree of uncertainty perfectly, we would expect a correlation of -1. Positive or low correlations would suggest that for the evaluated set of tasks, the developer is not good at separating effort estimates with high and low uncertainty.

2.4 Research Hypotheses

We designed the study to test the following four hypotheses.

Hypothesis 1. *Lessons-learned sessions improve estimation accuracy in comparison with on-the-job learning (lower median MRE).*

Hypothesis 2. *Lessons-learned sessions reduce estimation bias in comparison with on-the-job learning (median RE closer to 0).*

Hypothesis 3. *Lessons-learned sessions improve the realism of uncertainty estimation in comparison with on-the-job learning (OverConfidence closer to 0).*

Hypothesis 4. *Lessons-learned sessions improve the ability to separate low and high uncertainty estimates in comparison with on-the-job learning (CorrConfAcc closer to -1).*

2.5 The Research Design

2.5.1 Subjects

The subjects were recruited via a request for consultants that was sent to Norwegian consulting companies. The request specified a flexible range of time for which the consultants would be needed, along with the required education and expertise. Companies replied with *curricula vitae* of potential candidates and these were then screened to verify that they complied with the requirements. The subjects were required to at least have a Bachelor's degree in informatics (or equivalent) and familiarity with the technology (UML, Struts, JSP, Java, HTML, the Eclipse IDE, and MySQL) of the system on which they were supposed to complete development tasks.

In total, 20 software professionals (all male) were selected. The software professionals were paid close to ordinary fees for their work and asked to treat the development work as ordinary consultancy work. The software professionals did not know that developers other than themselves were asked to complete the tasks. The decision to include 20 software developers was mainly a practical, budget-related decision. It was also based on the belief that when studying learning processes, it would be better to let few developers complete more tasks than more developers fewer tasks.

2.5.2 Tasks

The experiment was conducted on the BESTweb system [4]. The BESTweb system is a Web-based system developed in Java using the Struts framework. The system supports the research on software cost and effort estimation through a database front-end client that gives access to information about journal and conference papers on software effort estimation. An essential part of the system's functionality is the identification of relevant papers through the use of categories related to, for example, type of estimation methods. As an illustration of the size of the software, the BESTweb software consists of about 50 classes and 3,000 lines of Java code. The BESTweb system can be accessed at www.simula.no/BESTweb.

All software professionals were instructed to estimate and complete the same five tasks in the same sequence. A brief description of the development tasks is given below.

- **Task 1**: Requires the addition of functionality to save a user's search query to persistent memory. In addition, the user's last search query must be displayed and reexecuted automatically when they next log on to the system.
- **Task 2**: Requires that the system be extended to handle an additional piece of data from an input file (in XML format) used to update the publications in the BESTweb system. The system already partially handles the data: If it encounters the presence of the data in the file, it warns the user that data are not supported. The developer is asked to add support for this data by extending the domain model, the GUI, and the search functionality.
- **Task 3**: Requires the developer to add functionality to the system that extends the manner in which cost and effort estimation metadata associated with each publication are dealt with, specifically, the ability to add publication categories and corresponding codes.
- **Task 4**: Requires the developer to add caching logic to the system so that if statistics for all the publications in the system are requested, the cached results are used (so as to decrease the computational load on the system).
- **Task 5**: The developers are asked to add functionality such that the users could delete existing publication codes from the system.

2.5.3 Experiment Process

The software professionals were allocated randomly, by the toss of a coin, to the Control or Learning group. The groups were the same size, 10 software professionals in each group. Each software professional was instructed to:

1. Read the description of the work process to be followed.
2. Complete an initial questionnaire that captured professional background and experience.

   FOR $i = 1$ TO 5 BEGIN

3. Read the specification of Task i.
4. Estimate the most likely effort needed to complete the specified task and assess the perceived uncertainty of that effort estimate, i.e., the subjective probabilities (confidence levels) of including the actual effort in the minimum-maximum intervals EI-1, EI-2, and EI-3.
5. Describe, in brief, the strategy used for the estimation and the assessment of uncertainty.
6. Complete Task i (design, program, test, and document).
7. Upon completion of the task, send in the task for acceptance testing, based on a predefined system acceptance test plan:

   a. If the test fails, fix the problem to submit the solution again (GOTO Step 6).
   b. If the test passes and the software professional belongs to the Learning group, receive the feedback on estimation accuracy and spend about 30 minutes in a lessons-learned session, i.e., filling out the fields on the right side of the vertical bar, as illustrated in Fig. 1.
   c. If the test passes and the software professional belongs to the Control group, then proceed.

   END /* for-loop */

8. Participate in a study debriefing session, where the purpose of the experiment was explained and the researchers tape-recorded responses to questions about estimation strategies and learning processes used.

The Control group process is meant to represents on-the-job learning, i.e., a process where no explicit steps are taken to increase learning. On-the-job learning is not a well-defined process and includes all types of processes where no explicit learning steps are taken. We expect, for example, that the developers use quite different learning processes during their estimation and programming work (the "on-the-job" part) in our study.

3 Results

3.1 Analysis of the Comparability of the Groups

A random allocation of as few as 20 subjects into two groups of the same size does not guarantee that the groups are similar with respect to estimation-relevant background and skill. Fortunately, we observed no essential differences, as illustrated in Table 1.

Table 2 extends the analysis of potential differences between the groups by comparing the performance on Task 1, i.e., the performance before the first exposure to the lessons-learned session.

It seems to be safe to claim, on the basis of Table 2, that both the effort and quality indicators of Task 1 were similar, which indicate that there were no large differences in programming skill. The higher mean MRE of the Control group is mainly due to one outlier, and the small difference in median values is probably more representative for the difference in estimation accuracy between the two groups.

A comparison of the uncertainty assessment-related values for IE-1, IE-2, and IE-3 for Task 1 (see Table 3), shows that there were no large difference here, either.

An informal analysis of the described strategies for estimating and assessing uncertainty (Step 5 of the

TABLE 1
Comparison of the Subjects' Backgrounds

Variable	Group	Mean	Lower Quartile	Median	Upper Quartile
Experience total (years)	Control	7,2	5,0	6,0	8,0
	Learning	7,0	4,0	5,5	10,3
Degree (1= Bachelors, 2= Masters)	Control	1,6	1,0	2,0	2,0
	Learning	1,9	2,0	2,0	2,0
Average grade in computer science courses	Control	2,0	1,7	2,0	2,4
	Learning	2,0	1,9	2,1	2,5
Self-assessed estimation competence (1=much worse than average, 5=much better than average)	Control	3,2	3,0	3,0	3,0
	Learning	2,9	3,0	3,0	3,0
Self-assessed average estimation error on previous tasks	Control	28%	19%	20%	43%
	Learning	25%	10%	23%	35%
Self-assessed programming skill (1=much worse than average, 5=much better than average)	Control	3,3	3,0	3,0	4,0
	Learning	3,2	3,0	3,0	4,0

TABLE 2
Group Comparison for Task 1: Effort, Quality, Estimation Accuracy, and Bias

Variable	Group	Mean	Lower Quartile	Median	Upper Quartile
Effort (work-hours)	Control	5,8	4,1	5,8	7,4
	Learning	6,0	4,1	4,8	9,1
Programming quality (mean number of failed acceptance tests)	Control	1,4	0,8	1,0	2,3
	Learning	1,5	0,0	1,0	3,0
Estimation accuracy (MRE)	Control	0,3	0,2	0,3	0,5
	Learning	0,4	0,2	0,4	0,6
Estimation bias (RE)	Control	0,11	-0,3	0,2	0,4
	Learning	0,05	-0,4	0,2	0,4

TABLE 3
Group Comparison for Task 1: Uncertainty Assessment

Variable	Group	Mean Confidence Level	Hit Rate	OverConfidence
Over-confidence IE-1	Control	52%	20%	1,6
	Learning	57%	20%	1,9
Over-confidence IE-2	Control	77%	50%	0,5
	Learning	80%	50%	0,6
Over-confidence IE-3	Control	96%	90%	0,7
	Learning	96%	90%	0,7

experiment process) used for Task 1 showed no essential differences. All of the developers relied on expert judgment supported by a separation of the task into subactivities, i.e., a bottom-up strategy. The processes used to assess uncertainty were poorly described and difficult to evaluate. This may indicate that the software professionals had no explicit strategy for uncertainty assessments, i.e., that the process was based more or less on expert judgment.

We also analyzed the information about the estimation processes and the reasons for accurate/inaccurate effort estimates on Task 1 with regard to references to similar tasks completed earlier. If, for example, some of the developers had conducted very similar tasks immediately before, difference learning progress may be expected. We found no such differences based on the estimation strategy descriptions provided for Task 1. None of the developers knew the software application before, but all had extensive experience with the technology applied.

In total, we find no essential differences between the Learning and the Control group harmful for our analysis purpose.

3.2 Estimation Accuracy and Bias

3.2.1 Hypothesis 1

Our first hypothesis states that the Learning group will improve the accuracy of their estimates more than the Control group, which will be indicated by the median MRE of the estimates being lower for Tasks 2-5 (the tasks following the first lessons-learned session). Fig. 2 shows that this was hardly the case. The median MREs of the Learning and the Control group across Tasks 2-5 are about the same (0.50 vs. 0.49). A Kruskal-Wallis test on difference in median MRE of the two groups for Tasks 2-5 gives $p = 0.88$.

Fig. 2 suggests that the MRE decreases over time for both groups, with the exception of Task 1, which may be easier than the other tasks. The actual estimation accuracy improvement is hard to evaluate properly without adjusting for the complexity of the tasks and is not the main topic of this study. In the absence of a proper control group, as is the case in typical field settings and poorly designed experiments, any observed improvement in MRE may falsely have been credited to the lessons-learned sessions. This supports our emphasis on introducing proper control groups in effect studies of this type. Notice also that a control group, as we use it here, enables much stronger cause-effect analyses than the introduction of a so-called baseline typically recommended for field studies (see, for example, [7]). While baseline data are typically collected before the process change takes place, our control group assumes a random allocation of process change and the parallel use of old and new processes. This means that any improvement that is measured in comparison to the baseline data must be able to isolate the effect of the process change from all other changes that took place in the relevant period. In our experience, this is a very difficult analysis with many challenges.

As can be seen in Fig. 2, the *variation* of MRE is not systematically lower in the Learning group. This result differs from the findings reported in [9], where the use of the Personal Software Process on students did lead to lower variance in estimation error (although not lower mean estimation error). This difference in results from those reported in [9] may have been caused by the use of formal effort estimation models when applying the Personal Software Process, as opposed to the use of judgment-based effort estimation process that was the case in our study. As pointed out in our review comparing models and expert judgment (see [5]), estimation models sometimes reduce the number of very large estimation errors and, consequently, the variance in estimation error. An increased level of consistency is also expected from the use of formal estimation models.

3.2.2 Hypothesis 2

Our second hypothesis states that the Learning group's estimates would be less biased, which would be indicated by the RE of the estimates provided by those in the Learning group being closer to zero than those in the Control group for Tasks 2-5. Fig. 3 shows that this was not the case. The total median RE of the Learning and the Control group turned out to be almost the same (-0.01 versus 0.00), which shows that the median effort estimate was unbiased in both groups. Unbiased effort estimates are not untypical when the tasks to be performed are small and payment is made on a per work-hour basis [6], as was the

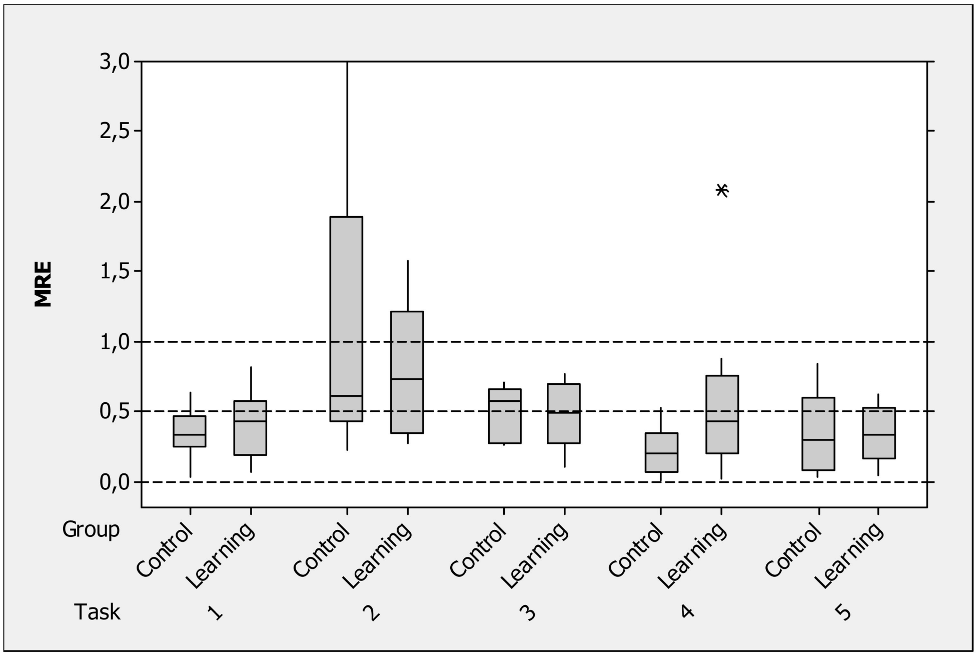


Fig. 2. Box plot of MRE (all tasks included).

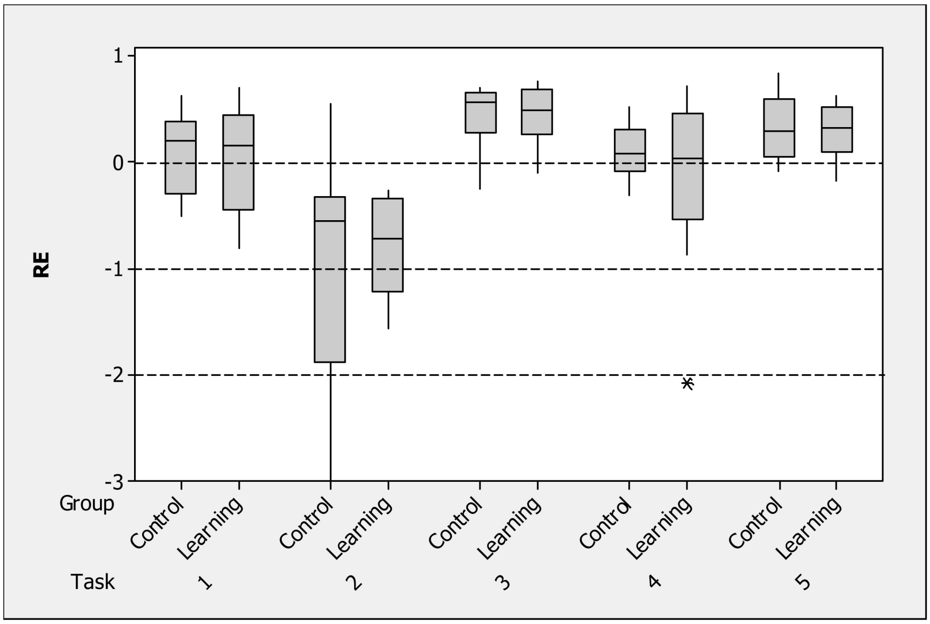


Fig. 3. Box plot of RE (all tasks included).

case in our study. A Kruskal-Wallis test on difference in median RE of the two groups for Tasks 2-5 gives $p = 1.0$.

Notice the zigzag pattern of the RE in Fig. 3. This pattern suggests that overoptimism on one task (e.g., Tasks 1) is easily followed by overpessimism on the next task (e.g., Tasks 2). If this suggestion is correct, it may be that there is a tendency to overreact to the experience of the task immediately before the one to be estimated.

3.3 Uncertainty Assessment

3.3.1 Hypothesis 3

Our third hypothesis states that the Learning group would improve the realism in their uncertainty assessment, which would be indicated by the OverConfidence of those in the Learning group being closer to zero than that of the Control group. We considered the number of observations to be too low for meaningful use of the OverConfidence measure on individual tasks; see the discussion in Section 2.3. That being so, the values presented are based on the distribution of OverConfidence of the developers on the combined set of Tasks 2-5.

Fig. 4 illustrates that the Learning group did not make more realistic uncertainty assessments than the Control group. Both groups were strongly overconfident. A Kruskal-Wallis test of difference in median OverConfidence values for IE-1, IE-2, and IE-3 gives $p = 0.6$, $p = 0.1$,

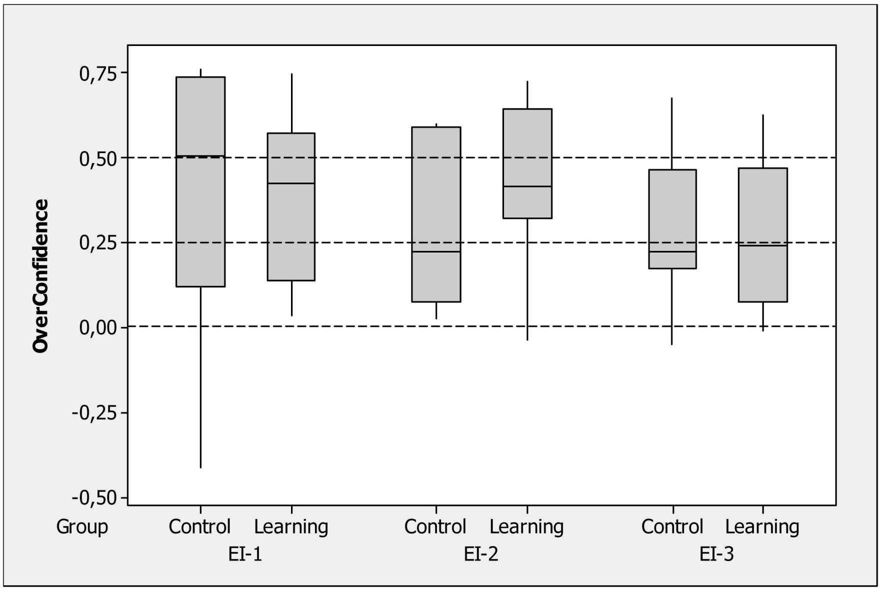


Fig. 4. Box plots of overconfidence (Tasks 2-5).

TABLE 4
Rank-Order Correlation between Confidence and MRE

Correlation between MRE and …	Group	Mean	Lower Quartile	Median	Upper Quartile
… confidence in EI-1	Control	-0,3	-0,7	-0,3	-0,1
	Learning	0,12	-0,3	0,4	0,5
… confidence in EI-2	Control	0,2	-0,3	0,3	0,7
	Learning	-0,04	-0,3	0,09	0,3
… confidence in EI-3	Control	0,09	-0,4	-0,05	0,7
	Learning	0,16	-0,3	0,18	0,7

and $p = 0.9$, respectively. In other words, the only difference in overconfidence between the groups with a low p-value, i.e., IE-2, is in favor of the Control group. If anything, the lessons-learned sessions seem to have resulted in an increase, rather than reduction, in the level of the participants' overconfidence in the accuracy of their estimates.

3.3.2 Hypothesis 4

Our fourth hypothesis states that the Learning group would be better able to separate low and high uncertainty estimates than the Control group, i.e., that those in the Learning group would be better than those in the Control group to provide low confidence levels when the estimation error (MRE) was high. We measure this ability through the Spearman rank-order correlation for each developer and each type of minimum-maximum interval (CorrConfAcc), i.e., three correlations per developer. The first relative uncertainty evaluation is on Task 2, where the confidence in the accuracy of Task 2 is compared with the confidence in the accuracy of Task 1. Given that the first relative uncertainty assessment is on Task 2 but involves Task 1, we include all five tasks in the rank-order correlation analysis. The presented median correlation of each group's minimum-maximum interval, e.g., the EI-1 interval, is consequently based on 10 individual correlation coefficients. The motivation and explanation of the CorrConfAcc measure are described in Section 2.3.

Descriptive statistics for the rank-order correlations are displayed in Table 4. We ranked the MRE and confidence values so that the highest values get the highest ranks. This means, as described earlier, that we should expect strong negative correlations if the developers are good at separating high and low uncertainty tasks, i.e., when a high confidence in the accuracy of an estimate correlates with a low estimation error (MRE). A perfect correlation between confidence and estimation error gives the value -1.

An analysis of the median values suggests that the Learning group is somewhat worse than the Control group for EI-1 and EI-3, but better for EI-2. However, the general conclusion is that the software professionals, regardless of group, were, in general, not very good at separating high and low uncertainty effort estimates. If anything, the Control group participants were slightly better. The ability to separate high and low uncertainty effort estimates depends on the similarity of the tasks [3], i.e., it is easier to separate low from high uncertainty estimates when the differences in uncertainty are larger. This means that our study does not exclude that the software professionals would perform better in other contexts. What we can claim based on the data in Table 4 is mainly that the lessons-learned session did not lead to improvement in the low ability to separate high and low uncertainty tasks in the studied context with five tasks completed on the same system.

4 Discussion

Our results show that the introduced lessons-learned session did not improve the learning in comparison with on-the-job learning. While our study may be the first on lessons-learned-based improvement of judgment-based effort estimation and uncertainty assessment, there are several other studies that report a similar lack of positive effects from lessons-learned sessions in other software project contexts, e.g., [24], [25], [26], and other project domain contexts, e.g., [27]. Shortcomings of software professionals' analyses of reasons for errors of estimation, such as the tendency to neglect indirect and contributing reasons and the tendency toward a biased attribution of causes (such as the tendency to claim that success factors are controlled by oneself, while failures are due to external events [28]), may contribute to this lack of learning from lessons-learned sessions.

Our results and those of previous studies on the use of lessons-learned sessions should, of course, not be used to claim that lessons-learned sessions will never result in improved estimation accuracy or more realistic assessment of uncertainty. There are many ways of designing lessons-learned sessions. It may well be that imposing other learning structures, other types of feedback, better training in advance of the lessons-learned sessions, or other measures, will yield different results. To understand how to improve lessons-learned sessions and other learning processes, it may be useful to look at possible reasons for the observed lack of improved learning from the lessons-learned session in our study. After all, to give up on learning from previous experience of estimation is not an attractive option.

To structure this discussion, we will analyze the following three types of possible reasons for the observed lack of improved learning:

- Limitation of the chosen study design (Section 4.1).
- Limitations of the lessons-learned process studied (Section 4.2).
- Observed problems related to learning from experience of the estimation of software development effort and the assessment of uncertainty (Section 4.3).

We will apply the qualitative data that we collected as part of our study to support this analysis. We collected this data (see description in Section 2) immediately after each estimate (where all the software professionals described their estimation processes) and as a part of the lessons-learned sessions (where the software professionals in the Learning group summarized their reasons for estimation errors and what they had learned). In addition, we will describe and apply the results from a follow-up experiment. In this experiment, other software professionals were asked to assess the uncertainty of the estimates provided in the main study. The purpose of the experiment was to examine whether there is a difference in realism between assessing the accuracy of one's own estimates and those of others, which would enable us to determine whether some of the observed problems with learning were due to the learning from one's own estimation performance as opposed to that of others.

4.1 Limitations of the Study Design

We assess that the major limitations of our study design are as follows:

- **Individual learning**. The software developers worked and learned individually, not as part of a development team. For example, two of the developers remarked in the debriefing session that in an ordinary work context they would have asked for some support by a colleague on at least one of the estimation tasks. Instead, they had to spend time to figure out the problem by themselves. This means that our results may mainly be representative for individual estimation learning and working, not so much for team learning. However, as argued earlier in this paper, individual learning is an essential component of group learning and, consequently, of relevance too.
- **Pseudorealistic context**. The software professionals were paid close to their ordinary fees, instructed to behave as close to normal as possible, and did not know that several others were completing the same tasks. However, the extensive logging (data about themselves, estimation strategy used, detailed time logging, etc.), the lessons-learned sessions (for the Learning group), and the fact that we as researchers were the clients meant that the situation was hardly perceived to be the same as in ordinary software development work. The software professionals expressed different opinions about the effect of the unusual process elements in the debriefing session. Our main impression from observing the work and analyzing the debriefing interviews is that, in spite of the unusual process elements, the estimation and programming work had the same challenges as ordinary programming work and, hence, the differences from ordinary work situations had only small, if any, effect on the learning processes. By observing the developers and talking with them in the debriefing sessions, we found no indications of low motivation or that they found the system or the tasks artificial.
- **Representativeness.** The high cost of hiring software professionals placed restrictions on the number of tasks that we could assign and the number of subjects that we could use. This means that we should be careful when extrapolating the results to tasks and situations that are different from those we have studied.

Notice that many of the unusual process elements of the study, e.g., the requirement to write down the lessons learned, may have led to a stronger focus on learning than in typical field situations. Thus, we find it difficult to see how individual, lessons-learned session-based learning would lead to improvements in typical field situations similar to the one we studied if it did not succeed here. Of course, this belief assumes that the software professionals were motivated to learn. We find that this assumption is likely to be true, judging from our observation of their behavior and the documents that were produced in the lessons-learned sessions. As an illustration, the software professionals wrote, on average, about 1,000 words each when describing lessons learned (about 200 words per lessons-learned session). The amount of lessons-learned work was impressive. It was also of good quality (more on this in the next section). Consequently, despite the limitations of the study, we interpret the findings of the study as describing a situation in which those in the Learning group really tried to use the learning sessions to learn, but nevertheless did not improve effort estimation accuracy more than the Control group, and were at least as overconfident in the accuracy of their effort estimates.

4.2 Limitations of the Learning Process

To examine what actually took place in the lessons-learned session, we examined the provided:

- reasons for good/poor estimation performance and uncertainty assessment (Section 4.2.1) and
- lessons learned relevant for estimation and uncertainty assessment and how the lessons learned may affect the estimation and uncertainty assessment of future tasks (Section 4.2.2).

4.2.1 Reasons for Accuracy/Inaccurate Estimates and Realistic/Unrealistic Uncertainty Assessments

The reasons described by the developers were as expected from previous studies on this topic; see [28] for an overview of typically provided reasons. The most common responses related to reasons for inaccurate estimates in the current study were:

- Spent too little time on effort estimation and uncertainty assessment work.
- Too little knowledge about the problem or the technology.
- Unexpected events or problems.
- Forgotten/strongly underestimated activities (such as documentation).
- Fewer and less severe problems than expected (led to overestimation).
- More complex task than expected.
- Task specification unclear or misunderstood the specification.
- Error corrections needed.
- Poor impact analysis of consequences of code changes.
- Design errors made.
- Incorrect assumptions about the code.

Typical responses related to reasons for accurate estimates were:

- Good impact analysis of consequences of code changes.
- Task similar to the one previously completed (on the same system).
- Good understanding.
- Luck.
- Simple task and no problems.

When we asked for reasons for realistic/unrealistic uncertainty assessments, we expected to be provided with reasons that were directly related to the uncertainty, such as insufficient risk analysis and lack of learning from the actual uncertainty of the previous effort estimates. Instead of these types of reason, the reasons for realistic/unrealistic uncertainty assessments provided by the developers were, almost without exception, related to the accuracy/inaccuracy of the effort estimate. This suggests that they believed that the dominant means to improve the realism of the uncertainty assessment was to improve estimation accuracy. The alternative, which would have been to change their level of confidence in the effort estimates, was hardly considered. An analysis of the confidence levels supports this suggestion. The analysis indicated that the software professionals changed their levels of confidence only a little in response to feedback about the accuracy of their previous estimate. If this observed uncertainty assessment improvement strategy is typical for software professionals, this may severely hinder learning. More studies should be conducted to examine the robustness of this finding and, if the improvement strategy is common among software professionals, how to change it to more appropriate strategies.

4.2.2 Lessons Learned

The main lessons learned related to estimation accuracy provided by those in the Learning group were the following:

- Spend more time on estimation work (including a more detailed impact analysis).
- Add more time for unknown events.
- Add more time for debugging, testing, and error correction.
- Add more time for documentation.
- Assess more carefully the ripple effects of a change.
- Add more time when the task involves changes at many different places in the code ("distributed" task).
- Read the requirement specification and system documentation more carefully and check the assumptions made.
- Pay special attention to the parts where the knowledge and experience is low.
- Trust one's instincts more.
- Expect unexpected problems.
- Control the use of effort better (do not do more than that which is required by the task specification).

All of these lessons learned were reasonable and have the potential to improve estimation accuracy. In that sense, our lessons-learned session process was a success in relation to the estimation of the most likely effort. The same cannot be said in relation to the lessons learned regarding the assessment of uncertainty. In spite of a direct instruction to summarize the lessons learned that were related to uncertainty assessment, we found, as noted in Section 4.2.1, no lessons learned that could be interpreted as practical changes in the process of assessing uncertainty.

We find it especially strange that the software professionals did not update their levels of confidence in response to feedback strongly suggesting overconfidence. Consider the following example: One software developer had assessed the probability of including the actual effort in the minimum-maximum interval IE-1 ([90 percent of estimate; 110 percent of estimate] to be about 50 percent of all of the first three tasks. The developer then received feedback that told him that none of the IE-1 intervals had included the actual effort. A proper lesson learned would then be for him to admit that he had a tendency toward overconfidence and he had to be less confident that he would to include the actual effort in IE-1 of future estimates. Instead, we observed that he typically responded with no or only a small adjustment and a strong emphasis on what he needed to do to achieve more accurate effort estimates. We discuss possible reasons for this unwillingness to update confidence levels in spite of accurate, relevant, and timely data on estimation error in [29], [30]. Possible reasons include: 1) The software professionals do not have proper, probabilistic mental models to enable proper learning and 2) there is a conflict between the developers' self-images of being predictable and skilled developers and providing realistic uncertainty assessments that suggest the opposite (the so-called "cognitive dissonance" effect). We discuss these two reasons in more detail in Section 4.3.

Given the intuitively meaningful lessons learned that were provided by the developers in relation to the estimation of most likely effort, why did the accuracy of their estimates not get better than that of the estimates of those in the Control group? Possible reasons for this include:

1. the software developers did not translate the identified lessons learned into practical actions;
2. the Control group developers learned as much as those in the Learning group without the lessons-learned session (the actual progress in learning, if any, is hard to evaluate due to the difference in complexity of the five tasks, but there may have been an improvement in estimation accuracy in both groups; see our discussion in Section 3.2.1);
3. there were negative consequences of the lessons learned that canceled out the positive ones; or
4. the lessons-learned sessions led the Learning group developers to believe that they had learned more than they actually had.

It is not easy to assess the importance of these possible reasons in isolation. The estimation strategy descriptions provided by the software professionals did not contain as much valuable information about the practical use of the lessons learned in the estimation work as we had hoped for. Nevertheless, it was clear that at least a few of the software professionals actually applied the lessons learned. As an illustration, when a developer stated that he needed to spend more time on the effort estimation work, we observed that he typically did so.

We counted the number of references to previous experience in the software professionals' description of the processes that they used for estimating and assessing uncertainty (Step 5 in the experiment process) and found no large differences in the number and/or types of references. This similarity in the amount of use of experience, together with the observed similarity in the improvement in the estimates of the Control and the Learning group, does provide some support for reason 2 above. Those in the Control group seemed to learn quite a lot from experience without spending time on lessons-learned sessions.

We also found some evidence in support of reason 3, namely, that the software professionals in the Learning group may have overreacted more strongly to the experience of the previous task. While the median absolute difference in RE between two tasks for those in the Control group was 0.6, the corresponding value of these in the Learning group was 0.8. A high difference in median RE is, for example, a result of going from strong underestimation to a strong overestimation of effort. This observation provides (weak) support that lessons learned may lead not only to positive effects, but also to stronger overreactions to previous experience. More studies are needed to examine whether this is true and, if so, the extent to which this is a typical effect of lessons-learned sessions or an effect that is caused by the particular context in our study. As reported in Section 1, however, our study is not the only one to observe possible negative effects of lessons-learned sessions. An important message, supported by our results, is, consequently, that meant-to-be-constructive learning processes may have negative consequences if not designed carefully. Preferably, such processes should be designed based on available knowledge about learning biases and limitations.

Reason 4 may be particularly interesting because it points at a difference between learning processes leading to improved effort estimation and, for example, improved programming skills. To estimate more accurately, a software developer not only has to learn from experience, but also assess the effect of the lessons learned on the future work and estimates. As an illustration, a software developer may through experience learn about limitations of the programming tools in use. This learning will most likely lead to improved programming efficiency and more predictable work on future tasks. The learning is, however, not sufficient for an improvement of the estimation accuracy. To improve the estimation accuracy, the developer also has to properly assess how much the learning about the development tool will lead to reduced effort and decreased uncertainty in use of effort on the next task. If the developer is overoptimistic about the reductions, the result may be a continued or even higher estimation inaccuracy and increased overconfidence in the effort estimates. This added step, i.e., the need for assessment of the effect of the learning, means that we should be careful about generalizing the lack of effect of lessons learned in estimation contexts to other learning contexts.

4.3 Learning Problems

Learning from experience is documented to be difficult both in software development [31] and other contexts. Hammond [32, p. 278] summarizes the situation: *"Yet in nearly every study of experts carried out within the judgment and decision-making approach, experience has been shown to be unrelated to the empirical accuracy of expert judgments."*

Reasons for the learning problems include the following:

- The "hindsight bias," e.g., the tendency to interpret a cause-effect relationship as more obvious after it has happened than before [33], [34].
- The tendency to confirm rules and disregard conflicting evidence, as illustrated in studies on human judgment [35], [36].
- The tendency to apply a "deterministic" instead of a "probabilistic" learning model. This ability to think in probability-based terms can, according to [37], hardly be derived from experience alone, but must be taught. Hammond [32] suggests that the ability to understand relationships in terms of probabilities instead of purely deterministic connections is important for correct learning in situations in which the level of uncertainty is high. When assessing the uncertainty of effort estimates, probabilistic thinking and learning models are essential. The observed problems with proper learning from feedback on uncertainty assessment in our study may be caused by a lack of training in the use of proper probabilistic learning models.
- The high amount of complex interconnected reasons for high or low estimation accuracy. In practice, we may need to include both system dynamics and game theory to understand the network of reasons for high or low estimation accuracy.

In many situations in which human judgment is used that has high uncertainty and unstable task relations, there are indications that even feedback of high quality, e.g., high-quality task-relation-oriented feedback, is not sufficient for learning [20], [38]. The underlying reason is that it is frequently difficult to transfer experience from one context to another. For this reason, it is important to recognize when there is nothing to learn from experience, as reported in the software estimation studies [31], [39].

On the basis of our previous experience, reported in [40], we believed that one essential reason for the sustained strong tendency toward overconfident uncertainty assessments could be related to the difference between assessing the uncertainty of one's own and others' effort estimates. When assessing one's own estimate of one's own work, there may be, for example, as pointed out earlier, learning problems related to the cognitive dissonance effect. This effect is, for example, present when the software professional tries to avoid a conflict between his image of himself as a skilled and predictable developer and historical data that suggest that the opposite is the case. A possible way to avoid this conflict is, for each new task, to assume either that one has learned much from previous experience or that the feedback about previous performance was, in some way, not relevant for the future. To test whether the problem of uncertainty assessment learning was related to this effect, we conducted a follow-up experiment at a seminar on effort estimation with software professionals as participants. If the assessment of other software professionals estimates, applying the same feedback and information, improves the learning, this may be used to improve the lessons-learned sessions. Alternatively, if the learning problems remain, this supports the belief that the lack of proper probabilistic learning models is the main obstacle to uncertainty assessment learning and that better training in probability and statistics are needed.

4.3.1 The Follow-Up Experiment

Participants. Eighty-three software professionals, about 90 percent male, who attended an estimation seminar at Simula Research Laboratory. These software professionals were similar to those in the main experiment with respect to length of experience and background.

Hypothesis. Assessment of the uncertainty of other developers' effort estimates based on the feedback about the estimation error of previous estimates will be more realistic than the assessment of the uncertainty of effort estimates of one's own work. (The realism of the uncertainty assessment will, as before, be indicated by the OverConfidence measure. The hypothesis implies that the participants in the follow-up experiment achieve an OverConfidence closer to zero than that of those in the main experiment.)

Process. Each of the 83 software professionals:

1. was randomly allocated to the estimates and feedback related to one of the 20 developers who participated in the main experiment.
2. read the description of what he was supposed to do, i.e., that he was to evaluate the uncertainty of effort estimates produced by a software developer on real software maintenance tasks.
3. read the description of the BESTweb system. (A short version of the description given to the developers in the main experiment.)
4. read the instruction on how to assess the uncertainty of the effort estimates. This instruction was the same as the one in the main experiment.
5. received a database with information about the estimation and uncertainty assessment of Tasks 1-3 as completed by the allocated developer in the main experiment. This database contained the task descriptions, the estimates of most likely effort, the actual use of effort values, and the estimation errors of all the three tasks. It did not contain the original effort uncertainty assessments.
6. received a description of Task 4 and the allocated developer's effort estimate of that task.
7. assessed the probability that the actual effort of the allocated developer would fall within IE-1 ([90 percent;110 percent] of the estimate), IE-2 ([60 percent;150 percent] of the estimate), and IE-3 ([50 percent;200 percent] of the estimate) for Task 4.
8. received the allocated developer's actual use of effort of Task 4.

TABLE 5
Mean OverConfidence of Others (Follow-Up Experiment Data) versus Own (Main Experiment Data) Estimates

Task	Assessment Type	IE-1	IE-2	IE-3
4	Others	-0,14	-0,21	-0,10
	Own	0,79	0,25	0,15
5	Others	0,71	0,39	0,09
	Own	1,75	0,98	0,28

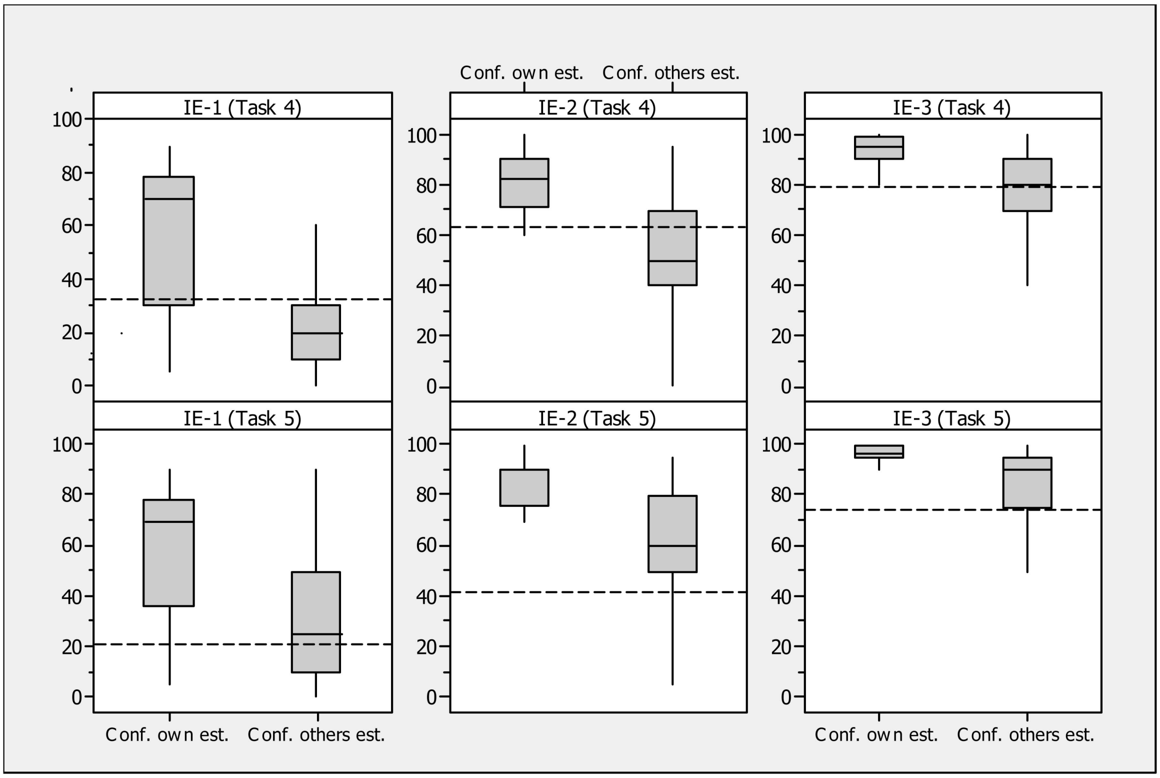


Fig. 5. Box plot of the IE-1, IE-2, and IE-3 confidence levels of Tasks 4 and 5.

9. received an updated version of the database, now including information about the first four tasks.
10. received a description of Task 5 and the allocated developer's effort estimate of that task.
11. assessed the probability that the actual effort would fall within IE-1 ([90 percent;110 percent] of the estimate), IE-2 ([60 percent;150 percent] of the estimate), and IE-3 ([50 percent;200 percent] of the estimate) for Task 5.

Table 5 compares the mean OverConfidence of the developers' assessment of the uncertainty of others (data from the follow-up experiment) and their own (data from the main experiment) estimates. As can be seen in Table 5, the uncertainty assessments of other developers' effort estimates, when supported by the same feedback of previous performance, are much more realistic (OverConfidence closer to zero) and less biased (both negative and positive OverConfidence values). While the uncertainty assessments in the main experiment, which were based on the developers' assessment of the accuracy of their own estimates, were strongly biased toward overconfidence, the assessments in the follow-up experiment, which were based on other developers' estimates, were more realistic, and even tended toward underconfidence for one of the tasks (Task 4).

Fig. 5 displays the distributions of confidence levels of Tasks 4 and 5 and all three uncertainty intervals. The horizontal, dotted line is the Hit rate for all 20 of the original developers for the different tasks and intervals. As before, the closer the median confidence level is to the Hit rate, the more realistic is the uncertainty assessment.

In the follow-up experiment, the developers focused only on performing well on assessing the uncertainty of effort estimates, while those in the main experiment tried to improve estimation accuracy and, perhaps even more, the performance of their programming work as well. It is for this reason, possibly, that the strong focus on uncertainty assessment in the follow-up experiment and not the difference between assessing the accuracy of one's own compared with other developers effort estimates explains the observed difference in realism. While this is a possible explanation and may contribute to the difference, we do not think that it is able to lead to the observed differences in realism alone. Those in the main experiment completed paid work, they spent more time on the uncertainty assessments, they knew that they would be evaluated on the realism of the uncertainty assessment (those in the follow-up experiment knew, on the other hand, that their assessments would not be evaluated), and they described and analyzed their uncertainty assessment performance and processes. This means that, even though those in the main experiment had a less narrow improvement focus, they also had more time, more support, and stronger motivation to perform well on the uncertainty assessment tasks.

While the software professionals in the main experiment were only male, the participants in the follow-up experiment included a small proportion (assessed, informally, to be about 10 percent) of female developers. We did not collect gender information and consequently could not analyze the effect of more female developers. There were, however, not enough female developers to explain the observed difference in realism between the main and the follow-up experiment. In addition, our previous, unpublished, analyses in similar estimation contexts suggest that the female population among software developers is not representative of the general female population. Female software developers seem to be just as overconfident as male developers.

The results from the follow-up experiment suggest, therefore, in our opinion, that the uncertainty assessment learning problems observed in the main experiment are not so much related to lack of proper probabilistic mental models, i.e., to lack of understanding of probabilities and how to adjust them, but more to psychological biases related to evaluation of one's own estimation accuracy performance. This suggests, for example, that learning processes related to performance improvement may be improved with the use of other developers.

5 Conclusions

Lessons-learned-based processes, such as postmortem analyses and project reviews, are frequently suggested for the purposes of improving processes [14], [41], [42]. Our results suggest that the type of individual lessons-learned processes examined in this study may have no, or even a negative, effect on the accuracy of estimates and the realism of assessments of the uncertainty of effort estimates in comparison with on-the-job learning due to, for example, learning biases related to the assessment of one's own estimation performance. Without better lessons-learned processes, there may be a substantial risk of wasting a lot of resources with no, or even a negative, effect in comparison with pure on-the-job learning. We recommend that there be a shift from the current tendency to propose lessons-learned processes that assume that it is sufficient to ask questions of the type "what can you learn from this" toward evidence-based learning processes that take into account, for example, the learning biases related to interpretation of one's own estimation performance.

Our study is conducted in the context of individual estimation and uncertainty assessment learning, and it is speculative to generalize to other software engineering contexts, e.g., group-based learning, from this study alone. There are, however, other studies that also report worrying results related to common lessons-learned processes (see Section 1). In total, we interpret this to strongly suggest that there is a need for improved lessons-learned processes and further studies. We believe that an appropriate study type for studies on learning processes is a controlled experiment with randomized treatment in close to field settings. Other study types may, however, also be applicable as long as the study design is able to convincingly attribute observed learning to the learning process or adjust for on-the-job learning that would have been present anyway.

Acknowledgments

The authors thank James Dzidek, Hans Christian Benestad, Stein Grimstad, Erik Arisholm, Dag Sjøberg, and other staff at Simula Research Laboratory for their contributions to this study.

Reference

[1] CompTIA, *Survey: Poor Communication Causes Most IT Project Failures. Inadequate Resource Planning, Unrealistic Deadlines Also Cited in CompTIA Study in Computerworld,* 2007.

[2] K. Moløkken and M. Jørgensen, "A Review of Software Surveys on Software Effort Estimation," *Proc. Int'l Symp. Empirical Software Eng.,* pp. 223-230, 2003.

[3] M. Jørgensen, K.H. Teigen, and K. Moløkken, "Better Sure than Safe? Over-Confidence in Judgement Based Software Development Effort Prediction Intervals," *J. Systems and Software,* vol. 70, nos. 1/2, pp. 79-93, 2004.

[4] M. Jørgensen and M. Shepperd, "A Systematic Review of Software Cost Estimation Studies," *IEEE Trans. Software Eng.,* vol. 33, no. 1, pp. 33-53, Jan. 2007.

[5] M. Jørgensen, "Estimation of Software Development Work Effort: Evidence on Expert Judgment and Formal Models," *Int'l J. Forecasting,* vol. 23, no. 3, pp. 449-462, 2007.

[6] M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," *J. Systems and Software,* vol. 70, nos. 1/2, pp. 37-60, 2004.

[7] W.S. Humphrey, *Introduction to the Personal Software Process.* Addison-Wesley, 1996.

[8] P. Abrahamsson and K.H. Kautz, "Personal Software Process: Classroom Experiences from Finland," *Proc. European Conf. Softwre Quality,* pp. 175-185, 2002.

[9] L. Prechelt and B. Unger, "An Experiment Measuring the Effects of Personal Software Process (PSP) Training," *IEEE Trans. Software Eng.,* vol. 27, no. 5, pp. 465-472, May 2000.

[10] V. Basili, H. Caldierea, and D. Rombach, "The Experience Factory," *Encyclopedia of Software Engineering,* J.J. Marciniak, ed., pp. 469-476, Wiley, 1994.

[11] S. Engelkamp, S. Hartkopf, and P. Brössler, "Project Experience Database: A Report Based on First Practical Experience," *Proc. Int'l Conf. Product Focused Software Development and Process Improvement,* pp. 204-215, 2000.

[12] F. Houdek, K. Schneider, and E. Wieser, "Establishing Experience Factories at Daimler-Benz: an Experience Report," *Proc. Int'l Conf. Software Eng.,* pp. 443-447, 1998.

[13] M. Jørgensen, D.I.K. Sjøberg, and R. Conradi, "Reuse of Software Development Experience at Telenor Telecom Software," *Proc. European Software Process Improvement Conf.,* pp. 10.19-10.31, 1998.

[14] A. Birk, T. Dingsøyr, and T. Stålhane, "Postmortem: Never Leave a Project without It," *IEEE Software,* vol. 19, no. 3, pp. 43-45, May/June 2002.

[15] T. Dingsøyr, "Postmortem Reviews: Purpose and Approaches in Software Engineering," *Information and Software Technology,* vol. 47, pp. 293-303, 2005.

[16] M.C. Ohlsson, C. Wohlin, and B. Regnell, "A Project Effort Estimation Study," *Information and Software Technology,* vol. 40, no. 14, pp. 831-839, 1998.

[17] W.K. Balzer, M.E. Doherty, and R. O'Connor, "Effects of Cognitive Feedback on Performance," *Psychological Bull.,* vol. 106, no. 3, pp. 410-433, 1989.

[18] P.G. Benson, "The Effects of Feedback and Training on the Performance of Probability Forecasters," *Int'l J. Forecasting,* vol. 8, no. 4, pp. 559-573, 1992.

[19] R.E. Stone and R.B. Opel, "Training to Improve Calibration and Discrimination: The Effects of Performance and Environmental Feedback," *Organizational Behavior and Human Decision Processes,* vol. 83, no. 2, pp. 282-309, 2000.

[20] N. Schmitt, B.W. Coyle, and L. King, "Feedback and Task Predictability as Determinants of Performance in Multiple Cue Probability Learning Tasks," *Organizational Behavior and Human Decision Processes,* vol. 16, no. 2, pp. 388-402, 1976.

[21] M. Schindler and M.J. Eppler, "Harvesting Project Knowledge: A Review of Project Learning Methods and Success Factors," *Int'l J. Project Management,* vol. 21, pp. 219-228, 2003.

[22] M. Cusomano and R. Selby, *Microsoft Secrets—How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People.* The Free Press, 1995.

[23] M. Jørgensen, "A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation," *Proc. First Int'l Workshop Software Productivity Analysis and Cost Estimation,* pp. 15-22, 2007.

[24] G. Pan, S.L. Pan, and M. Newman, "Information Systems Project Post-Mortem: Insights from an Attribution Perspective," *J. Am. Soc. for Information Science and Technology,* vol. 58, no. 14, pp. 2255-2268, 2007.

[25] K. Lyytinen and D. Robey, "Learning Failure in Information Systems Development," *Information Systems J.,* vol. 9, no. 2, pp. 85-101, 1999.

[26] D. Wastell, "Learning Dysfunctions in Information Systems Development: Overcoming the Social Defenses with Transitional Objects," *MIS Quarterly,* vol. 23, no. 4, pp. 581-600, 1999.

[27] M. Urban and A. Witt, "Self-Serving Biases in Group Member Attributions of Success and Failures," *J. Social Psychology,* vol. 130, no. 3, pp. 417-419, 1990.

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

[29] M. Jørgensen and K. Moløkken-Østvold, "Eliminating Over-Confidence in Software Development Effort Estimates," *Proc. Conf. Product Focused Software Process Improvement,* pp. 174-184, 2004.

[30] M. Jorgensen, K.H. Teigen, and K. Molokken, "Better Sure than Safe? Over-Confidence in Judgement Based Software Development Effort Prediction Intervals," *J. Systems and Software,* vol. 70, nos. 1/2, pp. 79-93, 2004.

[31] M. Jørgensen and D. Sjøberg, "The Importance of Not Learning from Experience," *Proc. European Software Process Improvement Conf.,* pp. 2.2-2.8, 2000.

[32] K.R. Hammond, *Human Judgement and Social Policy: Irreducible Uncertainty, Inevitable Error, Unavoidable Injustice.* Oxford Univ. Press, 1996.

[33] B. Fischhof, "Hindsight <> Foresight: The Effect of Outcome Knowledge on Judgement under Uncertainty," *J. Experimental Psychology: Human Perception and Performance,* vol. 1, pp. 288-299, 1975.

[34] D. Stahlberg et al. "We Knew It All Along: Hindsight Bias in Groups," *Organizational Behavior and Human Decision Processes,* vol. 63, no. 1, pp. 46-58, 1995.

[35] C.F. Camerer and E.J. Johnson, "The Process-Performance Paradox in Expert Judgment: How Can Experts Know So Much and Predict So Badly?" *Towards a General Theory of Expertise,* K.A. Ericsson and J. Smith, eds., pp. 195-217, Cambridge Univ. Press, 1991.

[36] D.M. Sanbonmatsu, A.A. Sharon, and E. Biggs, "Overestimating Causality: Attributional Effects of Confirmatory Processing," *J. Personality and Social Psychology,* vol. 65, no. 5, pp. 892-903, 1993.

[37] B. Brehmer, "In One Word: Not from Experience," *Acta Psychologica,* vol. 45, pp. 223-241, 1980.

[38] F. Bolger and G. Wright, "Assessing the Quality of Expert Judgment: Issues and Analysis," *Decision Support Systems,* vol. 11, no. 1, pp. 1-24, 1994.

[39] J. Shanteau, "Competence in Experts: The Role of Task Characteristics," *Organizational Behavior and Human Decision Processes,* vol. 53, no. 2, pp. 252-266, 1992.

[40] M. Jørgensen, "Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask," *IEEE Trans. Software Eng.,* vol. 30, no. 4, pp. 209-217, Apr. 2004.

[41] M. Cannon and A. Edmundson, "Failing to Learn and Learning to Fail (Intelligently): How Great Organizations Put Failure to Work to Innovate and Improve," *Long Range Planning,* vol. 38, pp. 299-319, 2005.

[42] B. Collier, T. Demarco, and P. Fearey, "A Defined Process for Project Post-Mortem Review," *IEEE Software,* vol. 13, no. 4, pp. 65-71, July 1996.

Magne Jørgensen is a professor at Simula Research Laboratory and the University of Oslo. He has previously worked as a software developer and a project manager and trained software professionals in the use of estimation models and expert judgment-based estimation processes. His research focus is on improving effort estimation and learning processes.

Tanja M. Gruschke is currently working toward the PhD degree at Simula Research Laboratory and University of Oslo with an interest in learning processes in software effort estimation. She is now working as a software developer at KnowIT Objectnet.