Expert Estimation of Software Development Work: Learning through Feedback

Magne Jørgensen
Dag Sjøberg
Simula Research Laboratory, Norway

Expert estimation seems to be the dominant effort estimation approach in software development work. This reliance on expert estimation is understandable considering the flexibility of expert estimation and the lack of empirical evidence that a change to model based effort estimation leads to higher accuracy. Surprisingly, several research studies report that the amount of experience is not a good indicator of estimation accuracy, even in situations where the experience is believed to be highly relevant, i.e., there seems to be problems connected with learning from estimation experience. This paper analyses potential reasons for this poor learning from experience. Based on these reasons we describe and validate four guidelines for improved learning: 1) Increase the motivation for learning estimation skills, 2) Reduce the impact from estimation learning biases, 3) Ensure a fit between estimation process and type of feedback, and 4) Provide learning situations. These guidelines extend the general software experience learning frameworks, e.g., the "experience factory" framework, with estimation specific learning guidelines. Through an experiment on software professionals we illustrate how a simple change in the estimation process based on the guidelines 2) and 3) made the estimation accuracy feedback significantly more useful for improvement of estimation performance through feedback.

1 Introduction

"How can experts know so much and predict so badly?" (Camerer and Johnson 1991)

Several empirical studies report that expert estimation of software development and maintenance effort is the dominant estimation approach (Heemstra and Kusters 1991; Hihn and Habib-Agahi 1991; Paynter 1996; Jørgensen 1997; Hill, Thomas et al. 2000; Kitchenham, Pfleeger et al. 2002). We were not able to find a single study that reported a dominance of model-based effort estimation. An important reason for this finding may be the flexibility of expert estimation. For example, while most formal estimation models require quantitative input on a pre-defined format, estimation experts may be able to apply whatever information available and produce, somehow, an effort estimate. Another reason for the dominance of expert estimation is that we lack substantial empirical evidence in favor of formal estimation models. We found fifteen studies comparing the estimation accuracy of expert estimates with that of model estimates for software development and maintenance work. Concerning the results of these studies, five were in favor of expert estimation (Kusters, Genuchten et al. 1990; Vicinanza, Mukhopadhyay et al. 1991; Mukhopadhyay, Vicinanza et al. 1992; Pengelly 1995; Kitchenham, Pfleeger et al. 2002), five found no difference (Heemstra and Kusters 1991; Ohlsson, Wohlin et al. 1998; Walkerden and Jeffery 1999; Bowden, Hargreaves et al. 2000; Lederer and Prasad 2000), and four were in favor of model-based estimation (Atkinson and Shepperd 1994; Jørgensen 1997; Myrtveit and Stensrud 1999; Jørgensen and Sjøberg 2002). An analysis of the designs of these fifteen studies indicates that expert estimation leads to higher estimation accuracy when the experts possess important domain or project knowledge not included in the estimation models. A comprehensive review of expert estimation studies, including a description of situations when we can expect expert-based estimates to be more accurate than model-based estimates is provided in (Jørgensen 2004b).

The accuracy of formal estimation models may be improved through an integration of the feedback[1] from completed projects. For example, a regression-based effort estimation model tailored to enhancement tasks of a particular application should, in theory, improve its estimation accuracy with increasing amount of completed enhancement projects and better understanding of relations between the variables. In practice, however, there are many reasons for this not to happen, e.g., the productivity of different software developers and teams may vary much, and, the processes and tools may change and the usefulness of historical data decreases.

While this model based estimation learning through feedback is explicit and the techniques to some extent well understood, the experts' estimation learning and processes are non-explicit and difficult to analyze. Consequently, there is a risk that the expert estimation feedback is not on a format, type and point-in-time that enables learning. The current state of learning estimation skills from on-the-job feedback and how to improve the learning are the topics of this paper. The remaining part is organized as follows:

- Section 2 examines how much software developers and maintainers improve the accuracy of their expert

[1] In the following, we interpret "feedback" as all types of information about software development processes and products enabling evaluation of and understanding of reasons for estimation (in)accuracy.

estimates from typical on-the-job estimation feedback and discusses the most important learning factors.

- Section 3 suggests and justifies four estimation learning guidelines. The guidelines are justified through the results from studies on software effort estimation studies and compared with results from other domains.
- Section 4 illustrates through the results from an experiment how the use of the guidelines can lead to improved estimation learning.
- Section 5 concludes the chapter.

2 Estimation Learning

Surprisingly, the experts' ability to learn from estimation experience seems to be disappointingly low for both maintenance and development of software applications. Jørgensen and Sjøberg (2002) report from a study of 54 software professionals' work on extending and maintaining large administrative applications. They found that the amount of application specific experience did not lead to more accurate assessments of maintenance risks, only to less unexpected maintenance problems. This finding does, of course, not suggest that experience *never* improves the estimation skills. Assume, for example, that two very similar maintenance tasks are conducted in sequence. Then, the feedback from the first task may be applied to increase the accuracy of the estimates and risk assessments of the next task. The reported results indicate, however, that the amount of such estimation situations is low and/or even then the feedback from the first task is difficult to apply properly for estimation purposes. A similar lack of learning from estimation feedback seems to be present for project leaders developing new applications. A study of six software project leaders' estimates over a period of three years showed no significance learning effect (Hill, Thomas et al. 2000).

The observed lack of learning from on-the-job feedback is not unique for software effort estimation, but has been observed in many other domains. For example, studies on experts' clinical and financial predictions show no correlation between experience and estimation skills. For overviews, see (Garb 1989) and (Blocher, Bouwman et al. 1997). Hammond (1996, p. 278) concludes the issue of experts' learning from estimation experience based on a high number of empirical studies: *"Yet in nearly every study of experts carried out within the judgement and decision-making approach, experience has been shown to be unrelated to the empirical accuracy of expert judgements."*

The two empirical studies reporting results on software development and maintenance estimation learning were not designed to investigate reasons for high/low learning from experience. This type of knowledge is, however, important to improve the learning. We should, therefore, try to identify and understand the factors that impact the learning. The following list of such factors are based on our examination of project experience reports and interviews with software professionals in three organizations (Jørgensen 1995; Jørgensen and Sjøberg 2001; Jørgensen and Sjøberg 2002):

- *Motivation*: There must be a motivation for collecting and learning from estimation feedback. The software developers and maintainers we interviewed, however, did not feel that estimation was "their main profession", although they all agreed that accurate estimates were important. Effort estimation was frequently perceived as an unpleasant task they were told to conduct now and then. In addition, the software professionals were typically not rewarded for high estimation accuracy. On the contrary, they were rewarded for inaccurate effort estimates, since an over-optimistic estimate and an over-confident assessment of estimation uncertainty were believed to be interpreted, by the project leaders, as high development skills. In an experiment (Jørgensen, Teigen et al. 2004), we found that then project leaders seemed to evaluate software developer as more skilled when providing over-confident assessments of estimation uncertainty compared with more proper descriptions of estimation uncertainty, even after the actual effort was known, i.e., even when they knew that the uncertainty assessments were incorrect. Consequently, a need for change in "code of practice", i.e., a change in the standards that the software professionals are measured by, may be beneficial to improve the estimation accuracy.
- *Awareness of the impact from human and situational biases*: There are many human and situational biases that hinder learning from experience. Kahneman et al (1982) provides an overview of several of them. A particularly interesting bias found in our examination of software project reports was related to how software professionals interpreted the reasons for high and low estimation accuracy (Jørgensen and Sjøberg 2001). It seems as if software professionals focused on *external* reasons, e.g., problems with sub-contractors, for inaccurate estimates, but attributed accurate estimates to their own estimation and project management skills. This bias may easily lead to a too high confidence in own estimation skills and a perception of no strong need for better learning processes. Similar results are found in other domains. For example Tan and Lipe (1997) found that: *"Those with positive outcomes (e.g., strong profits) are rewarded; justification or consideration of reasons as to why the evaluatee performed well are not necessary. In contrast, when outcomes are negative (e.g. losses suffered), justifications for the poor results are critical. .... Evaluators consider controllability or other such factors more when outcomes are negative than when they are positive."*
- *Easy evaluation and use of outcome feedback:* In (Jørgensen and Sjøberg 2001) we found a strong impact

from the effort estimate on the development and maintenance work. One software maintainer stated, for example: *"If we find in the detailed requirement specification phase that the effort estimates have been too low, we simplify the software solution."* Similar software development dynamics findings have been demonstrated in simulation experiments (Abdel-Hamid and Madnik 1983; Abdel-Hamid and Madnik 1986; Abdel-Hamid, Sengupta et al. 1999). This means that the estimation accuracy cannot always be evaluated as the deviation between estimated and actual effort, i.e., that the outcome feedback (which frequently is the only explicit estimation feedback received) is not always reliable information. If we cannot easily evaluate the quality of an effort estimate, it certainly is difficult to learn from the estimation experience.

- *Training opportunities*: To learn from experience there should be training opportunities and sufficient time should be allocated for learning (Ericsson and Lehmann 1996). However, the time a software professional spends on effort estimation seems to be low and fragmented. The time spent on writing experience reports, e.g., understanding the deviation between estimated and actual effort, may be even lower.
- *Timely feedback*: When estimating software tasks, the feedback is frequently only possible several weeks after it was provided. Then, the estimation assumptions and important events are easily forgotten or are subject to "hindsight biases" e.g., the tendency to interpret consequences and events as more obvious after it happen than before (Fischhof 1975; Stahlberg, Eller et al. 1995).
- *Causal models*: Our interviews with the software professionals gave that there was no explicit *causal* model to guide the estimation work. As reported in (Blocher, Bouwman et al. 1997) it is difficult to learn from experience without a causal model of the relation between important variables. Unfortunately, most current estimation models cannot be used as causal models, because they are based on correlations, not cause-effects, and do not including important task, application, developer and domain specific knowledge known by the software maintainers. For example, although there has been much research on the shape of the software "production function", i.e., function-based relation between input and output parameters, for several years, no agreement on its shape has been reached. Dolado (2001), for example, investigated the relationship between software size and effort on twelve data sets using regression analysis and genetic programming. He reported that it was hard to conclude on a relationship between effort and size, and that we could only expect moderately good results of size-based estimation models. Currently, most software development effort estimation models are size-based. Frequently, one individual's or one project's on-the-job experience does not enable the building of causal models, either. For example, we interviewed a number of project leaders about the reasons for inaccurate effort estimates (Jørgensen, Moen et al. 2002). One important reason for cost overruns was, according to the project leaders, incomplete requirement specifications. From the perspective of a project leader this opinion is easy to understand. A project may, for example, have had a lot of unexpected work on clarifying what the customer really wanted. However, when we compared the requirement specification information from the projects' experience reports with the cost estimation precision data in the experience database we found indications of the opposite, i.e., high estimation accuracy was more frequently connected with the *lack* of precise specifications. For example, one project experienced a major change in project scope in the middle of the project period and other unexpected project problems. Nevertheless, the estimation accuracy was very high. The experience report of that project indicated that the amount of delivered functionality and the quality was lower than intended when the cost was estimated, but still within the (high-level) requirement specification. The high estimation accuracy was, therefore, very much a result of the incomplete (flexible) requirement specification. The development of causal models relevant for estimation may therefore need analyses of a large set of projects, as well as deep analyses of single projects.
- *Standardization of the process*: Several widely used process improvement frameworks, e.g., the Capability Maturity Model, are based on the belief that standardization of the process lead to better predictability and less variation in use of effort. This belief is supported by opinions of project leaders we have interviewed, e.g., that the lack of a standardized process makes learning from previous experience difficult.
- *Code of practice:*

Learning estimation skills from experience is a hard task. The empirical studies and the learning hindrances discussed in this section suggest that we should not expect much estimation learning from typical on-the-job estimation feedback. Section 3 describes and validates several guidelines to improve this situation. In this chapter we focus on learning through estimation-related feedback.

3 Estimation Feedback and Process Guidelines

This section discusses and evaluates the following four learning guidelines: 1) Increase the motivation for learning estimation skills, 2) Reduce the impact from estimation learning biases, 3) Ensure a fit between estimation process and type of feedback, and 4) Provide learning situations. The selection of the guidelines is based on the discussion of learning factors in Section 2. Our intention is not to provide a complete set of guidelines, only to discuss what we believe are the most important guidelines for improvement of the *feedback-oriented estimation learning process*. We do, for example, not discuss the role of estimation courses and self-instruction through text-books in this section.

3.1 Increase the Motivation for Learning Estimation Skills

Two important process actions to increase the motivation for learning estimation skills are:

- *Evaluate the estimation accuracy of the software professionals*: Lederer and Prasad (1998) report that the factor with the highest impact on the estimation accuracy was the use of the estimation accuracy in the evaluation of the performance of the software professionals. Similarly, the studies (Weinberg and Schulman 1974; Jørgensen and Sjøberg 2001) found that inducing estimation accuracy as an important performance measure improved the estimation accuracy compared with situations where the projects were evaluated according to, e.g., time precision or quality. Higher motivation for learning from feedback may be only one out of many reasons for the measured improvement in estimation accuracy when evaluating the accuracy. Other reasons are, for example, more time spent on the estimation work and better project management to avoid effort-overrun. Nevertheless, the studies suggest that evaluation has an impact on the perceived importance of the estimation work. A perception of importance is, we believe, a pre-condition for learning estimation skills. It is important to be aware of that evaluation may have negative consequences, as well. Several human judgment studies suggest that a high motivation for estimation accuracy, for example when people feel personally responsible, perceive that the estimation task is very important or receive monetary rewards for accurate estimates, may *decrease* the estimation accuracy (Sieber 1974; Armstrong, Denniston Jr. et al. 1975; Cosier and Rose 1977). A possible explanation for this impact is that an increased awareness of being evaluated increases the level of so-called "dominant responses" (instincts) on cost of reflective responses (Zajonc 1965), i.e., a strong perception of evaluation leads to more instinct and less reflection. It is, for this reason, important to ensure that evaluation of software estimation accuracy does not lead to a strong pressure in the estimation situation, but instead to more emphasis on analysis and proper use of previous project and estimation experience.
- *Avoid estimation rewards different from accuracy*: As discussed in Section 2, some of the software professionals felt that they were rewarded with positive evaluation feedback when providing optimistic and over-confident estimates. The "estimation game" described in (Thomsett 1996) is a good example of this type of mixed rewards: "*Boss: Hi, Mary. How long do you think it will take to add some customer enquiry screens to the Aardvark System? Mary: Gee … I guess about six weeks or so. Boss: WHAAT?!!!! That long?!!! You're joking, right? Mary: Oh! Sorry. It could be done perhaps in four weeks*…." This type of situation, obviously, does not stimulate to learning. Another estimation accuracy evaluation practice reducing the learning is the lack of distinction between "estimated effort", "planned effort" and "bid", as reported in (Edwards and Moores 1994; Goodwin 1998; Jørgensen and Sjøberg 2001). The decisions on "bid", "planned effort" and "most likely effort" have conflicting goals and should be rewarded differently. A bid should, optimally, be low enough to get the job and high enough to maximize profit, the planned effort should enable a successful project and motivate to efficient work, and the estimate of the most likely effort should represent the most realistic use of effort. In order to stimulate estimation learning, we believe that the software developers' evaluation rewards should be on the estimated effort (not the planned effort or the bid) and the managers should be trained to elicit estimates so that that hidden evaluation goals (e.g., to "please the manager") are avoided.

3.2 Reduce the Impact from Estimation Learning Biases

There are several learning biases that should be accounted for when supporting the learning process. Among the most relevant biases are:

- *The "hindsight bias"*: Hindsight bias, i.e., the tendency to interpret cause-effect relationships as more obvious after it happened than before (Fischhof 1975; Stahlberg, Eller et al. 1995), is difficult to avoid. For example, it may not be sufficient to inform about it and instruct people to avoid it (Slovic and Fischhoff 1977).
- *The tendency to confirm rules and disregard conflicting evidence*: Empirical results suggests that learning processes easily lead to the mode of "confirming theories on how to complete the project", rather than

"reject incorrect hypotheses and assumptions" (Brehmer 1980; Koehler 1991).

- *The over-confidence in own estimates*: A strong tendency to ignore previous feedback and persist in over-confidence in own estimates have been observed in software development and other domains (Koriat, Lichtenstein et al. 1980; Connolly and Dean 1997; Yaniv and Foster 1997; Jørgensen, Teigen et al. 2004). We discuss reasons for this and how to overcome when presenting the learning experiment in Section 4.

Probably, there is no easy method to avoid these learning biases. Although important, it probably not sufficient to make software professionals aware of the learning biases. In addition to increased awareness, software professionals should be stimulated to documenting the estimation assumptions and criticizing own interpretation of learning (Silverman 1992; Brenner, Koehler et al. 1996).

A change of the estimation process itself may also reduce an impact on the learning biases. For example, it seems as if estimation processes that use historical data more explicit, e.g., top-down, analogy-based estimation processes, are less biased than those relying on an "inside-view" of the software to be developed, e.g., bottom-up, work-breakdown structure-based estimation processes (Moløkken 2002). Similar results are reported in (Kahneman and Lovallo 1993).

3.3 Ensure a Fit Between the Estimation Process and Type of Feedback

Several human judgment studies show a disappointing lack of estimation improvement from outcome feedback, i.e., feedback relating the actual outcome to the estimated outcome (Balzer, Doherty et al. 1989; Benson 1992; Stone and Opel 2000). The only software effort estimation study on this topic we were able to find (Ohlsson, Wohlin et al. 1998) show the same result. That study reports no estimation improvement from outcome estimation feedback. In fact, the only situation were outcome feedback is reported to improve the estimation accuracy seems to be when the estimation tasks are "dependent and related" and the estimator initially was under-confident, i.e., underestimated her/his own knowledge (Subbotin 1996).

The lack of improvement from outcome feedback should be no large surprise, since there is a lack of fit between the information needed when estimating and the type of feedback provided. For example, it is not easy to apply the feedback *"the effort estimate on your previous project was 30% too low"* when estimating a new project. Although not very useful for improvement of estimation accuracy, outcome feedback seems to be useful for the learning of estimation uncertainty assessments (Stone and Opel 2000). In the uncertainty assessment situation there is a better fit between the estimation process and the type of feedback. For example, if all projects similar to that to be estimated were estimated within +/- 50% of the actual effort, it is likely that the new project's estimate are likely to be within this uncertainty interval. Unfortunately, the typical estimation uncertainty process is not designed to make maximum benefit from the estimation outcome feedback. We discuss how to improve the fit between uncertainty estimation and feedback in Section 4.

To improve the estimation accuracy, several studies suggest that "task relation oriented feedback", i.e., feedback on how different events and variables were related to the actual use of effort, are required (Schmitt, Coyle et al. 1976; Balzer, Doherty et al. 1989; Benson 1992; Stone and Opel 2000). Possible methods to provide this type of feedback is to use experience reports, e.g., as part of an "experience factory", or "post mortem analyses" (Basili, Caldierea et al. 1994; Houdek, K et al. 1998; Jørgensen, Sjøberg et al. 1998; Engelkamp, Hartkopf et al. 2000; Birk, Dingsøyr et al. 2002).

3.4 Provide Learning Situations

As discussed in Section 2, it is not easy to learn from on-the-job estimation feedback alone. The time spent on estimation is low and fragmented, it is not easy to derive cause-effects based on few projects, and the time from estimation to feedback is frequently long. We therefore recommend that software organizations introduce the "cost engineer" role (a person dedicated to analysis of all the estimation work) and "deliberate learning" (artificial estimation situations designed for specific learning purposes) to improve the individual and organizational learning. Our recommendations are based on the following argumentation:

- The importance of causal domain models for learning is emphasized in (Bolger and Wright 1994). See also discussion in Section 2. A person dedicated to analyzing and summarizing estimation relevant information, i.e., the cost engineer, is likely to do a better cause-effect estimating analysis than software professionals dedicating a low and fragmented part of their time to estimation. This argumentation is in line with the "experience factory" organizational principles described in (Basili, Caldierea et al. 1994).
- It has been shown that immediate feedback is able to strongly improve the estimation accuracy (Bolger and Wright 1994; Shepperd, Fernandez et al. 1996). Interestingly, Shepperd et al. (1996) also found that when the feedback is rapid, people with low confidence start to under-estimate their own performance, maybe to ensure that they will not be disappointed, i.e., there may be situations where the feedback can be *too* rapid too stimulate realistic estimates. On-the-job estimation situations can seldom provide immediate feedback, and artificial learning situations may be necessary. We recommend that software organizations introduce so-called "deliberate training" as described by Ericsson and Lehmann (1996), i.e., *"individualized training activities especially designed by a coach or teacher to improve specific aspects of an individual's*

performance through repetition and successive refinement". Ericsson and Lehmann (1996) found that it was not the amount of experience but the amount of "deliberate training" that determined the level of expertise, e.g., the estimation skills. This importance of designed training situations is also supported by the results described in (Camerer and Johnson 1991), where it was found that while training had an effect on estimation accuracy, amount of experience had almost none. We suggest that software companies provide estimation training opportunities through their database of completed projects. An estimation training session could, for example, include estimation of completed projects based on the information available at the point-of-estimation applying different estimation processes. This type of estimation training has several advantages in comparison with the traditional estimation training:

- Individualized feedback, the actual effort and the project's experience report, can be received immediately after completion of the estimates.
- The effect of not applying checklists and other estimation tool can be investigated on one's own estimation processes.
- The validity of own estimation experience can be examined on different types of projects, i.e., projects much larger than those estimated earlier.
- Reasons for forgotten activities or underestimated risks can be analyzed immediately, while the hindsight bias is weak.
- The tendency to be over-confidence can be acknowledged and compensated for, given proper coaching and training projects.

As far as we know, there are no reported studies of organizations conducting estimation training in line with our suggestions. However, the results from other studies, in particular those summarized in (Ericsson and Lehmann 1996), strongly support that this type of training should complement the traditional estimation courses and on-the-job "learning from experience".

4 Experiment: Application of the Guidelines

The experiment[2] described in this section illustrates how the application of guideline 2 (reduce the impact of estimation learning biases) and 3 (ensure a fit between the estimation process and the feedback) can lead to learning improvements. The estimation topic studied is the learning of effort estimation uncertainty assessment through effort prediction (minimum-maximum) intervals. An effort prediction interval consists of a minimum and maximum value and a connected confidence level (Moder, Phillips et al. 1995). For example, a software maintainer may be 90% confident that the actual effort of a maintenance task will be between 20 and 40 work-hours. Then, the 90% effort prediction interval is [20, 40] work-hours.

4.1 Background

There is strong empirical evidence that the prediction intervals, on average, are much too narrow to reflect the stated confidence level. For example, we observed the effort prediction intervals of a number of industrial software projects and found that the hit rate, i.e., the proportion of tasks were the actual effort was inside the minimum-maximum values, was only about 30% (Jørgensen, Teigen et al. 2004). Most of these projects did not state the confidence level, but, obviously, most meaningful interpretations of minimum-maximum values imply hit rates higher than 30%. Similarly, we found that the 90% prediction intervals of student projects had a hit rate of only 62%[3]. Similar prediction interval hit rates were found by Conolly and Dean (1997) on computer science students on programming tasks. The students and the software professionals had previous experience developing and estimating software tasks. Why were they not able to learn from this experience to achieve better, or at least unbiased, assessment of the uncertainty of their effort estimates?

Analysing the learning process we found two important reasons for lack of learning: 1) The feedback they had received on previous tasks was difficult to apply in the next uncertainty assessment situation, i.e., there was not a good fit between the estimation process and the type of feedback, 2) The situational and human biases stimulated the software professionals to remain over-confident, i.e., not to learn from previous estimation outcome. The following example illustrates this. A software maintainer is asked to provide an effort estimate and a 90% confidence effort prediction interval for a new task. The maintainer estimates the most likely effort to be 20 work-hours based on experience with similar tasks, but is not sure how to decide on a 90% confidence effort prediction interval. Based on a search on previous estimation performance data on similar tasks, the software maintainer finds that the estimation accuracy varied from -10% to +60%, i.e., from 10% too high to 60% too low estimates. The median estimation deviation was a 30% too low estimate. Further, the maintainer knows that the previous 90% confidence prediction intervals included only 60% of the actual values, i.e., the

[2] The results from the experiment were presented at the International conference on Project Management (ProMAC), Singapore, 2002. The discussion in this paper extends the discussions of the previous conference paper with more analysis of reasons for (not) learning from feedback.

[3] Just before the students estimated their projects, they attended an estimation lecture emphasizing the typical too narrow minimum-maximum intervals. In spite of this lecture, the intervals were much too narrow to reflect a 90% confidence level.

maintainer receives the feedback that he/she has previously been strongly over-confident. Even with all this estimation feedback, which is much more than typically received, the maintainer does not know how to determine the minimum and maximum value for a 90% confidence level. The maintainer knows that she/he, on average, should increase the width of the prediction interval, but not how much. A formal calculation of the effort prediction interval is statistically complex[4] and a recent study suggest that statistically computed effort prediction intervals applies the uncertainty information inefficiently (Jørgensen 2004a). The lack of useful estimation feedback means that the maintainer has to base the uncertainty assessment on task knowledge, e.g., through construction of scenarios leading to different use of effort. Several studies on time-estimation, see (Buehler, Griffin et al. 1997) for an overview, suggest that this strategy tends to lead to over-confidence and limited learning from experience.

In addition, there may be strong situational biases that leads to over-confidence, i.e., to too narrow prediction intervals. For example, the maintainer knows that she/he will be perceived as more skilled when providing narrow prediction intervals. To be perceived as a skilled maintainer may be perceived as much more important than a proper prediction interval. In particular, this is likely to happen if the maintainer expects that there will be no evaluation on the prediction interval.

In total, the feedback does not support the uncertainty assessment, i.e., prediction interval, process and the situational biases stimulate to over-confidence, i.e., to too narrow prediction intervals.

The following experiment illustrates how a simple change in the uncertainty estimation process can lead to a large impact on the usefulness of the estimation outcome feedback and on the bias towards over-confidence.

4.2 Experiment Design

The TRADITIONAL prediction interval elicitation process, see for example the PERT process described in (Moder, Phillips et al. 1995) is:

1. Estimate the most likely effort.
2. Select a confidence level for the prediction interval to be estimated in Step 3. Typically, confidence levels reflecting a 90% confidence or more are selected, e.g., the confidence level described as "almost certain".
3. Estimate the minimum and maximum effort so that the probability of including the actual effort is believed to equal the confidence level.

As discussed earlier this process does not enable an efficient use of estimation feedback and typically leads to over-confidence, i.e., to too narrow effort prediction intervals. An ALTERNATIVE uncertainty assessment process is the following:

1. Estimates the most likely effort.
2. Calculate the minimum and maximum effort as fixed proportions of the most likely effort, e.g., apply the effort prediction interval guidelines from NASA ((1990)minimum effort = 50% and maximum effort = 200% of the most likely effort).
3. Estimate the confidence level, i.e., the probability that the actual effort is between the minimum and maximum effort.

Formally, the difference between these two processes is not large However, from an estimation learning perspective there are important differences. The described problems with the TRADITIONAL elicitation process may, to some extent, be removed by the ALTERNATIVE process. Assume that a project leader is asked to provide the confidence level of a 50%-200% effort prediction interval of a software project and that there exists a number of similar, previously completed, projects that have been estimated applying this 50%-200% fixed-width effort prediction interval. A simple analysis of the previous prediction intervals gives, for example, that 70% of those intervals included the actual effort. Consequently, given that the history is relevant, the expected probability that the actual effort is included by the new 50%-200% prediction interval is about 70%. As opposed to the TRADITIONAL process, no sophisticated statistical calculations are needed. Further, since the minimum and maximum effort values are calculated "mechanically", it may also be easier to be realistic about the probability that the actual effort is inside the interval, i.e., the over-confidence may be reduced. While minimum and maximum values provided by oneself, as in the TRADITIONAL process, may be used to signalize skill, those provided automatically may reduce the feeling of prediction interval ownership and consequently increase the focus on realism.

To test the effect of the ALTERNATIVE uncertainty elicitation process we paid twenty-nine software developers/project managers of a Norwegian, e-commerce software development organization to participate in an experiment. The participants were, randomly, divided into two groups: TRADITIONAL and ALTERNATIVE. The participants of the TRADITIONAL group used the TRADITIONAL uncertainty

[4] The statistical definition of a prediction interval, assuming a single random sample of *n* independent observations from a normally distributed population, is Christensen, R. (1998). Analysis of variance, design and regression. Applied statistical methods, Chapman & Hall/Crc.: $\bar{y} \pm t(conf, df) \cdot s \cdot \sqrt{1 + \frac{1}{n}}$, *where* $\bar{y}$ is the sample mean, *t(conf,df)* is the t-value for confidence level *conf* and *df* degrees of freedom, and *s* is the standard deviation of the sample.

elicitation process and those of the ALTERNATIVE group used the ALTERNATIVE uncertainty elicitation process. All participants should estimate the most likely effort and the uncertainty of the estimate of 30 software estimation tasks. The estimation work was completed in a Microsoft Excel spreadsheet. There was no interaction between the participants during the estimation work. The experiment was divided into two parts:

1) TRAINING: Instructions and training in use of the estimation spreadsheet and how to provide effort prediction intervals. As training tasks the effort and the effort prediction intervals of 10 real-world software development projects were estimated based on an "experience database" of 5 similar projects. Feedback was given after each estimate. When the estimates were completed, the software developers were asked to analyse and reflect on their own performance, in particular the correspondence between confidence level and hit rate.
2) ESTIMATION: Estimation of 30 software enhancements tasks previously conducted in a large telecom company. Task 1 was estimated based on an "experience database" including 5 previously completed tasks; Task 2 was estimated based on the 5 tasks and feedback, i.e., the actual effort, on the Task 1 estimate; Task 3 was estimated based on the 5 tasks and the feedback on the Tasks 1 and 2 estimates, and so on.

The sequence of the 30 tasks was randomized for all the software developers, i.e., the software developers estimated the tasks in different sequences. The alternative, to let all the developers estimate the tasks in the same sequence, was avoided, because then the chosen sequence might have resulted in tasks with increasing or decreasing estimation complexity. Consequently, a measured improvement might have been caused by simpler tasks to be estimated, not necessarily a real learning from experience.

There are some factors that are different from a realistic estimation task. For example, in most realistic cases the estimators would have more relevant information about the application, e.g., the complexity of the particular part to be changed, about the particular developer who carried out the task and about the necessary implementation, testing and integration activities. This means that we cannot expect very accurate predictions of the actual effort, i.e., the effort prediction intervals probably get wider than those of real software task estimation situations. The learning opportunities, due to fast feedback, may be better compared with real life estimation tasks.

4.3 Results

Two of the participants who followed the TRADITIONAL prediction interval estimation process did not understand the task of estimating the prediction intervals properly and were removed from the data set, leaving 13 software professionals in the TRADITIONAL group and 14 in the ALTERNATIVE group. To study the progress of the correspondence between hit rates and confidence, we divided the 30 tasks into three sets: Task 1-10, Task 11-20 and Task 21-30. There was no large difference in the accuracy of the estimate of the most likely use of effort, i.e., there was no difference in estimation skills of the two groups. Both groups had a median deviation between the estimated and actual effort of about 50%.

Table 1 shows the differences in the correspondence between mean hit rates (Hit rate) and mean confidence levels (Conf.) for the two elicitation processes.

Table 1: Mean Hit Rate vs. Mean Confidence Level

	Task 1-10		Task 11-20		Task 21-30	
Group	Hit rate	Conf.	Hit rate	Conf.	Hit rate	Conf.
TRADITIONAL	0.64	0.90	0.70	0.90	0.81	0.90
ALTERNATIVE	0.67	0.73	0.69	0.71	0.73	0.71

The participants who followed the TRADITIONAL process approached very slowly an acceptable correspondence between hit rate and confidence level. Even the Task 21-30 data show a tendency to over-confidence! The participants who followed the ALTERNATIVE process, on the other hand, had a close to perfect correspondence between mean hit rate and confidence level on all groups of tasks.

In real software development work, there are rarely as much as 20-30 similar tasks that can be used as basis for new estimates. This means that the performance of the 10 first estimation tasks, based on a few earlier tasks, may be the most important criterion for comparing the two estimation process. The experiment indicates that the benefit of the ALTERNATIVE elicitation process is at its largest on the first tasks. This strongly suggest that the effort prediction elicitation change from TRADITIONAL to ALTERNATIVE based on the learning guidelines 2) and 3) was effective. The effect of the change is clearly visible also on an individual level, see Table 2. Pairs of hit rate and mean confidence level with deviation larger than or equal to 0.2 are printed in bold.

Table 2: Hit Rate vs. Mean Confidence Level (Individual Data)

		Training tasks		Task 1-10		Task 11-20		Task 21-30	
Person	Process	Hit rate	Conf.	Hit rate	Conf.	Hit rate	Conf.	Hit rate	Conf.
T1	TRADITIONAL	**0,5**	**0,9**	**0,7**	**0,9**	0,8	0,9	1,0	0,9
T2	TRADITIONAL	0,9	0,9	0,9	0,9	0,9	0,9	**0,7**	**0,9**
T4	TRADITIONAL	**0,5**	**0,9**	**0,6**	**0,9**	**0,6**	**0,9**	0,9	0,9
T5	TRADITIONAL	**0,5**	**0,9**	**0,4**	**0,9**	**0,7**	**0,9**	**0,5**	**0,9**
T6	TRADITIONAL	0,8	0,9	**0,7**	**0,9**	**0,5**	**0,9**	1,0	0,9
T7	TRADITIONAL	**0,7**	**0,9**	**0,7**	**0,9**	**0,7**	**0,9**	0,8	0,9
T8	TRADITIONAL	**0,6**	**0,9**	**0,6**	**0,9**	**0,6**	**0,9**	0,8	0,9
T9	TRADITIONAL	0,8	0,9	0,8	0,9	0,8	0,9	1,0	0,9
T10	TRADITIONAL	**0,7**	**0,9**	**0,4**	**0,9**	0,9	0,9	**0,7**	**0,9**
T11	TRADITIONAL	**0,3**	**0,9**	**0,6**	**0,9**	0,9	0,9	0,8	0,9
T12	TRADITIONAL	**0,4**	**0,9**	**0,6**	**0,9**	**0,3**	**0,9**	0,8	0,9
T13	TRADITIONAL	**0,2**	**0,9**	**0,7**	**0,9**	**0,7**	**0,9**	**0,7**	**0,9**
A1	ALTERNATIVE	0,9	0,9	**0,5**	**0,7**	**0,9**	**0,7**	0,7	0,7
A2	ALTERNATIVE	**0,7**	**0,9**	**0,6**	**0,8**	0,8	0,7	0,9	0,9
A3	ALTERNATIVE	0,7	0,8	0,6	0,6	0,6	0,5	**0,8**	**0,6**
A4	ALTERNATIVE	n.a.	n.a	**0,5**	**0,9**	**0,6**	**0,9**	0,8	0,9
A5	ALTERNATIVE	**0,8**	**1,0**	**0,6**	**0,9**	**0,5**	**1,0**	0,9	0,9
A6	ALTERNATIVE	0,8	0,8	0,6	0,7	**0,5**	**0,7**	0,7	0,6
A7	ALTERNATIVE	**0,6**	**0,9**	0,8	0,8	0,7	0,8	0,8	0,8
A8	ALTERNATIVE	0,9	0,8	0,7	0,8	0,8	0,7	0,8	0,7
A9	ALTERNATIVE	0,8	0,8	0,7	0,7	0,7	0,7	0,7	0,7
A10	ALTERNATIVE	1,0	1,0	0,8	0,8	0,7	0,7	0,6	0,7
A11	ALTERNATIVE	**1,0**	**0,8**	0,7	0,7	0,5	0,6	0,6	0,6
A12	ALTERNATIVE	**0,9**	**0,7**	**0,9**	**0,6**	**0,7**	**0,5**	**0,8**	**0,5**
A13	ALTERNATIVE	0,8	0,8	0,7	0,7	**0,9**	**0,7**	**0,4**	**0,7**
A14	ALTERNATIVE	0,9	0,8	0,7	0,6	**0,8**	**0,6**	0,7	0,6

As expected, most of the hit rates of those who followed the TRADITIONAL process were too low to reflect a 90% confidence level, i.e., the prediction intervals were strongly over-confident. The mean difference between the hit rate and the 90% confidence level was -0.2, i.e., strongly biased. The participants who followed the ALTERNATIVE process, however, had sometimes a too high and sometimes a too low confidence level. The mean difference between the hit rate and the confidence was -0.02, i.e., there was no bias. Unbiased estimates means, amongst others, that the ALTERNATIVE process enables substantial accuracy improvement through averaging estimates from different prediction sources (Armstrong 2001). The expected improvement through combination of prediction intervals from the TRADITIONAL approach is much lower.

There are limitations regarding the generality of our results, for example:

- Only one level of confidence (TRADITIONAL process) and one interval width (ALTERNATIVE process) were studied. These levels were chosen because they were typical (90% confidence) or recommended (50%-200% prediction intervals). It is, nevertheless, possible that other confidence levels and interval widths would lead to other results. For example, it is possible that the selection of a 70% confidence level would remove the bias towards over-confidence applying the TRADITIONAL process, and that an 80%-120% prediction interval would lead to over-confidence applying the ALTERNATIVE process. In other words, the improvements we found may depend on the chosen levels. For example, Seaver et al. (1978) found no large difference between the two processes for the "inter-quartile" interval, i.e., the interval that should reflect a 50% confidence level.
- The estimation process of the participants in the experiment was different from their normal process. Normally, they would know more about the task, the development environment, etc. There is, therefore, a need for a replication of the study in more realistic estimation situations. We have now started a study comparing the TRADITIONAL and ALTERNATIVE process in real industrial software development estimation processes.

5 Conclusion

On-the-job estimation learning opportunities seems to be low and explain the observed lack of estimation learning from experience. To improve the learning situation we propose the following four learning guidelines:

1) Increase the motivation for learning estimation skills, 2) Reduce the impact from estimation learning biases, 3) Ensure a fit between estimation process and type of feedback, and 4) Provide learning situations. Based on results from studies in software development and other domains we argue that the proposed learning guidelines are capable of improving the feedback-oriented estimation learning. To follow the guidelines means, for example, that software organizations should increase their knowledge on how they motivate, evaluate and reward estimation, and what type of information the software professionals actually use when estimating effort. Based on an analysis the type of feedback and the learning situations provided should be designed, e.g., following the steps we describe in Section 4. While much of the learning actions to be implemented may be simple, e.g., to use estimation accuracy as an evaluation criterion, other changes may require more reflections and research, e.g., how to design "deliberate estimation training" without too high costs.

Acknowledgement: Thanks to professor in psychology at the University of Oslo, Karl Halvor Teigen, for his useful suggestions and interesting discussions.

References

Abdel-Hamid, T. K. and Madnik, S. E. (1983). "The dynamics of software project scheduling." Communications of the ACM **26**(5): 340-346.

Abdel-Hamid, T. K. and Madnik, S. E. (1986). "Impact of schedule estimation on software project behavior." IEEE Software **3**(4): 70-75.

Abdel-Hamid, T. K., Sengupta, K., et al. (1999). "The impact of goals on software project management: An experimental investigation." MIS Quarterly **23**(4): 531-555.

Armstrong, J. S. (2001). Combining Forecasts. Principles of forecasting: A handbook for researchers and practitioners. J. S. Armstrong, Kluwer Academic Publishers**:** 417-439.

Armstrong, J. S., Denniston Jr., W. B., et al. (1975). "The use of the decomposition principle in making judgments." Organizational-Behavior-and-Human-Decision-Processes. **14**(2): 257-263.

Atkinson, K. and Shepperd, M. (1994). Using function points to find cost analogies. European Software Cost Modelling Meeting, Ivrea, Italy.

Balzer, W. K., Doherty, M. E., et al. (1989). "Effects of cognitive feedback on performance." Psychological Bulletin **106**(3): 410-433.

Basili, V., Caldierea, H., et al. (1994). The Experience Factory. Encyclopedia of Software Engineering. J. J. Marciniak, Wiley**:** 469-476.

Benson, P. G. (1992). "The effects of feedback and training on the performance of probability forecasters." International Journal of Forecasting **8**(4): 559-573.

Birk, A., Dingsøyr, T., et al. (2002). "Postmortem: Never leave a project without it." IEEE Software **19**(3): 43-45.

Blocher, E., Bouwman, M. J., et al. (1997). "Learning from experience in performing analytical procedures." Training Research Journal **3**: 59-79.

Bolger, F. and Wright, G. (1994). "Assessing the quality of expert judgment: Issues and analysis." Decision support systems **11**(1): 1-24.

Bowden, P., Hargreaves, M., et al. (2000). "Estimation support by lexical analysis of requirements documents." Journal of Systems and Software **51**(2): 87-98.

Brehmer, B. (1980). "In one word: Not from experience." Acta Psychologica **45**: 223-241.

Brenner, L. A., Koehler, D. J., et al. (1996). "On the evaluation of one-sided evidence." Journal of Behavioral Decision Making **9**(1): 59-70.

Buehler, R., Griffin, D., et al. (1997). "The role of motivated reasoning in optimistic time predictions." Personality and social psychology bulletin **23**(3): 238-247.

Camerer, C. F. and Johnson, E. J. (1991). 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, Cambridge University Press**:** 195-217.

Christensen, R. (1998). Analysis of variance, design and regression. Applied statistical methods, Chapman & Hall/Crc.

Connolly, T. and Dean, D. (1997). "Decomposed versus holistic estimates of effort required for software writing tasks." Management Science **43**(7): 1029-1045.

Cosier, R. A. and Rose, G. L. (1977). "Cognitive conflict and goal conflict effects on task performance." Organizational Behaviour and Human Performance **19**(2): 378-391.

Dolado, J. J. (2001). "On the problem of the software cost function." Information and Software Technology **43**(1): 61-72.

Edwards, J. S. and Moores, T. T. (1994). "A conflict between the use of estimating and planning tools in the management of information systems." European Journal of Information Systems **3**(2): 139-147.

Engelkamp, S., Hartkopf, S., et al. (2000). Project Experience Database: A Report Based on First Practical Experience. PROFES, Oulu, Finland, Springer-Verlag: 204-215.

Ericsson, K. A. and Lehmann, A. C. (1996). "Expert and exceptional performance: Evidence of maximal adaptation to task constraints." Annual Review of Psychology **47**: 273-305.
Fischhof, B. (1975). "Hindsight <> foresight: The effect of outcome knowledge on judgement under uncertainty." Journal of Experimental Psychology: Human Perception and Performance **1**: 288-299.
Garb, H. N. (1989). "Clinical judgment, clinical training, and professional experience." Psychological bulletin **105**(3): 387-396.
Goodwin, P. (1998). Enhancing judgmental sales forecasting: The role of laboratory research. Forecasting with judgment. G. Wright and P. Goodwin. New York, John Wiley & Sons**:** 91-112.
Hammond, K. R. (1996). Human judgement and social policy. New York, Oxford University Press.
Heemstra, F. J. and Kusters, R. J. (1991). "Function point analysis: Evaluation of a software cost estimation model." European Journal of Information Systems **1**(4): 223-237.
Hihn, J. and Habib-Agahi, H. (1991). Cost estimation of software intensive projects: A survey of current practices. International Conference on Software Engineering, IEEE Comput. Soc. Press, Los Alamitos, CA, USA: 276-287.
Hill, J., Thomas, L. C., et al. (2000). "Experts' estimate of task duration in software development projects." International Journal of Project Management **18**: 13-21.
Houdek, F., K, S., et al. (1998). Establishing Experience Factories at Daimler-Benz An Experience Report. International Conference on Software Engineering, Kyoto, Japan: 443-447.
Jørgensen, M. (1995). "An empirical study of software maintenance tasks." Journal of Software Maintenance **7**: 27-48.
Jørgensen, M. (1997). An empirical evaluation of the MkII FPA estimation model. Norwegian Informatics Conference, Voss, Norway, Tapir, Oslo: 7-18.
Jørgensen, M. (2004a). "Regression Models of Software Development Effort Estimation Accuracy and Bias." To appear in: Empirical Software Engineering.
Jørgensen, M. (2004b). "A Review of Studies on Expert Estimation of Software Development Effort." Journal of Systems and Software **70**(1-2): 37-60.
Jørgensen, M., Moen, L., et al. (2002). Combining Quantitative Software Development Cost Estimation Precision Data with Qualitative Data from Project Experience Reports at Ericsson Design Center in Norway. Proceedings of the conference on empirical assessment in software engineering, Keele, England, Keele University.
Jørgensen, M. and Sjøberg, D. I. K. (2001). "Impact of effort estimates on software project work." Information and Software Technology **43**(15): 939-948.
Jørgensen, M. and Sjøberg, D. I. K. (2002). "Impact of experience on maintenance skills." Journal of Software Maintenance and Evolution: Research and practice **14**(2): 123-146.
Jørgensen, M., Sjøberg, D. I. K., et al. (1998). Reuse of software development experience at Telenor Telecom Software. European Software Process Improvement Conference (EuroSPI'98), Gothenburg, Sweden: 10.19-10.31.
Jørgensen, M., Teigen, K. H., et al. (2004). "Better Sure than Safe? Overconfidence in Judgment Based Software Development Effort Prediction Intervals." Journal of System and Software **70**(1-2): 79-93.
Kahneman, D. and Lovallo, D. (1993). "Timid choices and bold forecasts: A cognitive perspective on risk taking." Management Science **39**(1): 17-31.
Kahneman, D., Slovic, P., et al. (1982). Judgment under uncertainty: Heuristics and biases. Cambridge, United Kingdom, Cambridge University Press.
Kitchenham, B., Pfleeger, S. L., et al. (2002). "An empirical study of maintenance and development estimation accuracy." Journal of Systems and Software **64**(1): 57-77.
Koehler, D. J. (1991). "Explanation, imagination, and confidence in judgment." Psychological bulletin **110**(3): 499-519.
Koriat, A., Lichtenstein, S., et al. (1980). "Reasons for confidence." Journal of Experimental Psychology: Human Learning and Memory **6**(2): 107-118.
Kusters, R. J., Genuchten, M. J. I. M., et al. (1990). "Are software cost-estimation models accurate?" Information and Software Technology **32**(3): 187-190.
Lederer, A. L. and Prasad, J. (1998). "A causal model for software cost estimating error." IEEE Transactions on Software Engineering **24**(2): 137-148.
Lederer, A. L. and Prasad, J. (2000). "Software management and cost estimating error." Journal of Systems and Software **50**(1): 33-42.
Moder, J. J., Phillips, C. R., et al. (1995). Project management with CPM, PERT and precedence diagramming. Wisconsin, U.S.A, Blitz Publishing Company.
Moløkken, K. (2002). Expert estimation of Web-development effort: Individual biases and group processes (Master Thesis). Department of Informatics, University of Oslo.
Mukhopadhyay, T., Vicinanza, S. S., et al. (1992). "Examining the feasibility of a case-based reasoning model for software effort estimation." MIS Quarterly **16**(2): 155-171.
Myrtveit, I. and Stensrud, E. (1999). "A controlled experiment to assess the benefits of estimating with analogy

and regression models." IEEE Transactions on Software Engineering **25**: 510-525.
Ohlsson, N., Wohlin, C., et al. (1998). "A project effort estimation study." Information and Software Technology **40**(14): 831-839.
Paynter, J. (1996). Project estimation using screenflow engineering. International Conference on Software Engineering: Education and Practice, Dunedin, New Zealand, IEEE Comput. Soc. Press, Los Alamitos, CA, USA: 150-159.
Pengelly, A. (1995). "Performance of effort estimating techniques in current development environments." Software Engineering Journal **10**(5): 162-170.
Reifer, D. J. (1990). "ASSET-R: A function point sizing tool for scientific and real-time systems." Journal of Systems and Software **11**(3): 159-172.
Schmitt, N., Coyle, B. W., et al. (1976). "Feedback and task predictability as determinants of performance in multiple cue probability learning tasks." Organizational Behaviour and Human decision processes. **16**(2): 388-402.
Seaver, D. A., von Winterfeld, D., et al. (1978). "Eliciting subjective probability distributions on continuous variables." Organizational Behavior and Human Performance **21**: 352-379.
Shepperd, J. A., Fernandez, J. K., et al. (1996). "Abandoning unrealistic optimism: Performance estimates and the temporal proximity of self-relevant feedback." Journal of Personality and Social Psychology **70**(4): 844-855.
Sieber, J. E. (1974). "Effects of decision importance on ability to generate warranted subjective uncertainty." Journal of Personality and Social Psychology **30**(5): 688-694.
Silverman, B. G. (1992). "Judgment error and expert critics in forecasting tasks." Decision Sciences **23**(5): 1199-1219.
Slovic, P. and Fischhoff, B. (1977). "On the psychology of experimental surprises." Journal of Experimental Psychology: Human Perception and Performance **3**(4): 544-551.
Stahlberg, D., Eller, F., et al. (1995). "We knew it all along: Hindsight bias in groups." Organizational Behaviour and Human Decision Processes **63**(1): 46-58.
Stone, E., R and Opel, R. B. (2000). "Training to improve calibration and discrimination: The effects of performance and environmental feedback." Organizational Behaviour and Human Decision Processes **83**(2): 282-309.
Subbotin, V. (1996). "Outcome feedback effects on under- and overconfident judgments (general knowledge tasks)." Organizational Behaviour and Human Decision Processes **66**(3): 268-276.
Tan, H.-T. and Lipe, M. G. (1997). "Outcome effects: the impact of decision process and outcome controllability." Journal of Behavioral Decision Making **10**(4): 315-325.
Thomsett, R. (1996). "Double Dummy Spit and other estimating games." American Programmer **9**(6): 16-22.
Vicinanza, S. S., Mukhopadhyay, T., et al. (1991). "Software effort estimation: An exploratory study of expert performance." Information systems research **2**(4): 243-262.
Walkerden, F. and Jeffery, R. (1999). "An empirical study of analogy-based software effort estimation." Journal of Empirical Software Engineering **4**(2): 135-158.
Weinberg, G. M. and Schulman, E. L. (1974). "Goals and performance in computer programming." Human Factors **16**(1): 70 - 77.
Yaniv, I. and Foster, D. P. (1997). "Precision and accuracy of judgmental estimation." Journal of Behavioral Decision Making **10**(1): 21-32.
Zajonc, R. B. (1965). "Social facilitation." Science **149**(Whole No. 3681): 269-274.