A Review of Studies on Expert Estimation of Software Development Effort

M. JØRGENSEN
magne.jorgensen@simula.no
Simula Research Laboratory,
P.O.Box 134, 1325 Lysaker, Norway

Abstract
This paper provides an extensive review of studies related to expert estimation of software development effort. The main goal and contribution of the review is to support the research on expert estimation, e.g., to ease other researcher's search for relevant expert estimation studies. In addition, we provide software practitioners with useful estimation guidelines, based on the research-based knowledge of expert estimation processes. The review results suggest that expert estimation is the most frequently applied estimation strategy for software projects, that there is no substantial evidence in favour of use of estimation models, and that there are situations where we can expect expert estimates to be more accurate than formal estimation models. The following twelve expert estimation "best practice" guidelines are evaluated through the review: 1) Evaluate estimation accuracy, but avoid high evaluation pressure, 2) Avoid conflicting estimation goals, 3) Ask the estimators to justify and criticize their estimates, 4) Avoid irrelevant and unreliable estimation information, 5) Use documented data from previous development tasks, 6) Find estimation experts with relevant domain background and good estimation records, 7) Estimate top-down and bottom-up, independently of each other, 8) Use estimation checklists, 9) Combine estimates from different experts and estimation strategies, 10) Assess the uncertainty of the estimate, 11) Provide feedback on estimation accuracy and development task relations, and, 12) Provide estimation training opportunities. We found supporting evidence for all twelve estimation principles, and provide suggestions on how to implement them in software organizations.

Keywords: Software development, effort estimation, expert judgment, project planning

1 Introduction

Intuition and judgment – at least good judgment – are simply analyses frozen into habit and into the capacity for rapid response through recognition. Every manager needs to be able to analyze problems systematically (and with the aid of the modern arsenal of analytical tools provided by management science and operations research). Every manager needs also to be able to respond to situations rapidly, a skill that requires the cultivation of intuition and judgment over many years of experience and training. (Simon 1987)

In this paper we summarize empirical results related to expert estimation of software development effort. The primary goal and contribution of the paper is to support the research on software development expert estimation through an extensive review of relevant papers, a brief description of the main results of these papers, and the use of these results to validate important expert estimation guidelines. Although primarily aimed at other researchers, we believe that most of the paper, in particular the validated guidelines, are useful for software practitioners, as well.

We apply a broad definition of expert estimation, i.e., we include estimation strategies in the interval from unaided intuition ("gut feeling") to expert judgment supported by historical data, process guidelines and checklists ("structured estimation"). Our main criteria to categorize an estimation strategy as expert estimation is that the estimation work is conducted by a person recognized as an expert on the task, and that a significant part of the estimation process is based on a non-explicit and non-recoverable reasoning process, i.e., "intuition". Most estimation processes have both intuitive and explicit reasoning elements, as reported in the business forecasting study described in (Blattberg and Hoch 1990). In fact, even formal software development estimation models may require expert estimates of important input parameters (Pengelly 1995), i.e., require non-explicit and non-recoverable reasoning. Estimation strategies where a formal model is at the core of the estimation process are, however, not the topic of this paper.

There are relatively few studies discussing software development effort expert estimation. For example, a search for estimation papers in the journals IEEE Transactions on Software Engineering, Journal of Systems and Software, Journal of Information and Software Technology, and Journal of Empirical Software Engineering resulted in exactly 100 papers on software effort or size estimation[1]. Of these, only 17 (17%) include analyses or discussions of expert estimation; (Kusters 1990; Taff, Borchering et al. 1991; van Genuchten and Koolen 1991;

[1] Search conduced March 2002.

Betteridge 1992; Goodman 1992; Abdel-Hamid, Sengupta et al. 1993; Londeix 1995; Hughes 1996b; Höst and Wohlin 1997; Lederer and Prasad 1998; Ohlsson, Wohlin et al. 1998; Chulani, Boehm et al. 1999; Myrtveit and Stensrud 1999; Verner, Overmyer et al. 1999; Walkerden and Jeffery 1999; Mizuno, Kikuno et al. 2000; Jørgensen and Sjøberg 2001a). Similarly, while there have been several surveys of software development effort estimation models, e.g., (Mohanty 1981; Boehm 1984; Hihn and Habib-Agahi 1991b; Fairley 1992; Heemstra 1992; Walkerden and Jeffery 1997; Boehm and Sullivan 1999; Boehm, Abts et al. 2000; Briand and Wieczorek 2002), we found only one survey on expert estimation research results (Hughes 1996a). Fortunately, there are many relevant studies on expert estimation in other domains, e.g., medicine, business, psychology, and project management. To evaluate, understand, and extend the software development expert estimation results, we therefore try to transfer selected expert estimation research results from other domains.

We have structured the large amount of empirical results around a discussion and empirical validation of twelve "best practice" expert estimation principles. The selection of those principles was based on three sources: (1) What we have observed as best expert estimation practice in industrial software development projects, (2) The list of 139 forecasting principles described in (Armstrong 2001d), and, (3) The nine software estimation principles described in (Lederer and Prasad 1992). The selected twelve estimation principles do, of course, not cover all aspects of software development effort expert estimation. They provide, however, a set of principles that we believe are essential for successful expert estimation. Table 1 describes the topics and main result of each section of this paper.

Table 1: Contents of Paper

Section	Description of Topic	Main Results
2	Frequency of use of expert estimation.	Expert estimation is the dominant strategy when estimating software development effort.
3	Performance of expert estimation in comparison with estimation models.	The design of the empirical studies comparing expert and model-based software development effort estimate seems to have had a strong impact on the results. It is not possible to conclude that expert estimation or estimation model, in general, are more accurate. However, expert estimates seems to be more accurate when there are important domain knowledge *not* included in the estimation models, when the estimation uncertainty is high as a result of environmental changes not included in the model, or when simple estimation strategies lead to relatively accurate estimates.
4	Reduce situational and human biases.	Empirical validation of the expert estimation principles: 1. Evaluate estimation accuracy, but avoid high evaluation pressure. 2. Avoid conflicting estimation goals. 3. Ask estimators to justify and criticize their estimates. 4. Avoid irrelevant and unreliable estimation information. 5. Use documented data from previous development tasks. 6. Find estimation experts with relevant domain background and good estimation records.
5	Support the estimation process.	7. Estimate top-down and bottom-up, independently of each other. 8. Use estimation checklists. 9. Combine estimates from different experts and estimation strategies. 10. Assess the uncertainty of the estimate.
6	Provide feedback and training opportunities.	11. Provide feedback on estimation accuracy and task relations. 12. Provide estimation training opportunities.
7	Conclusions and further research.	All 12 principles are based on empirical evidence. There is, however, still a need for more knowledge about how to apply them in various software estimation situations.

2 Frequency of Use of Expert Estimation

Published surveys on estimation practice suggest that expert estimation is the dominant strategy when estimating software development effort. For example, the study of software development estimation practice at Jet Propulsion Laboratory reported in (Hihn and Habib-Agahi 1991a) found that 83% of the estimators used "informal analogy" as their primary estimation techniques, 4% "formal analogy" (defined as expert judgment based on documented projects), 6% "rules of thumb", and 7% "models". The investigation of Dutch companies described in (Heemstra and Kusters 1991) conclude that 62%, of the organizations that produced software development estimates, based the estimates on "intuition and experience" and only 16% on "formalized estimation models". Similarly, a survey conducted in New Zealand (Paynter 1996) reports that 86% of the responding software development organizations applied "expert estimation" and only 26% applied "automated or manual models" (an organization could apply more than one method). A study of the information systems development department of a large international financial company (Hill, Thomas et al. 2000) found that *no* formal software estimation model was used. Jørgensen (1997) reports that 84% of the estimates of software development projects conducted in a large Telecom company were based on expert judgment, and Kitchenham et al. (2002) report that 72% of the project estimates of a software development company were based on "expert judgment". In fact, we were not able to find any study reporting that *most* estimates were based on formal estimation models. The estimation strategy categories and definitions are probably not the same in the different studies, but there is nevertheless strong evidence to support the claim that expert estimation is more frequently applied than model-based estimation. This strong reliance on expert estimation is not unusual. Similar findings are reported in, for example, business forecasting, see (Remus, O'Connor et al. 1995; Winklhofer, Diamantopoulos et al. 1996).

There may be many reasons for the reported low use of formal software development effort estimation models, e.g., that software organizations feel uncomfortable using models they do not fully understand. Another valid reason is that, as suggested in our survey in Section 3, we lack substantial evidence that the use of formal models lead to more accurate estimates compared with expert estimation. The strong reliance on the relatively simple and flexible method of expert estimation is therefore a choice in accordance with the method selection principle described in "Principles of Forecasting" (Armstrong 2001c, p. 374-375): *"Select simple methods unless substantial evidence exists that complexity helps. ... One of the most enduring and useful conclusions from research on forecasting is that simple methods are generally as accurate as complex methods."* However, even if we had substantial evidence that the formal models led to, on average, more accurate estimates, this may not be sufficient for widespread use. Todd and Benbasat (2000), studying people's strategies when conducting decisions based on personal preferences, found that a decision strategy also must be easier to apply, i.e., demand less mental effort, than the alternative (default) decision strategy to achieve acceptance by the estimators. Similarly, Ayton (1998) summarizes studies from many domains where experts were resistant to replace their judgments with simple, more accurate decision rules.

3 Performance of Expert Estimation in Comparison with Estimation Models

We found fifteen different empirical software studies comparing expert estimates with estimates based on formal estimation models. Table 2 briefly describes the designs, the results and the, from our viewpoint, limitations of the studies in a chronological sequence. We do not report the statistical significance of the differences in estimation accuracy, because most studies do not report them, and because a meaningful interpretation of significance level requires that: 1) A population (of projects, experts, and estimation situations) is defined, and, 2) A random sample is selected from that population. None of the reported studies define the population, or apply random samples. The samples of projects, experts and estimation situations are better described as "convenience samples". We use the term "expert" (alternatively, "software professional" or "project leader") in the description of the estimators, even when it is not clear whether the estimation situation, e.g., experimental estimation task, enables the expert to apply his/her expertise. Consequently, experts may in some of the studies be better interpreted as novices, even when the participants are software professionals and not students.

Table 2: Software Studies on Expert Estimation of Effort

Nr	References	Designs of Studies	Results and Limitations
1	(Kusters, Genuchten et al. 1990)	Experimental comparison of the estimation accuracy of 14 project leaders with that of estimation models (BYL and Estimacs) on 1 finished software project.	The project leaders' estimates were, on average, more accurate than the estimation model. Limitations: 1) The experimental setting, 2) The estimation models were not calibrated to the organization.
2	(Vicinanza, Mukhopadhyay et al. 1991)	Experimental comparison of the estimation accuracy of 5 software professionals with that of estimation models (function points and COCOMO) on 10 finished software projects.	The software professionals had the most and least accurate estimates, and were, on average, more accurate than the models. Limitation: 1) The experimental setting. 2) The project information was tailored to the estimation models, e.g., no requirement specification was available, and 3) The estimation models were not calibrated to the organization.
3	(Heemstra and Kusters 1991)	Questionnaire based survey of 597 Dutch companies.	The organizations applying function points-based estimation models had the same estimation accuracy as those *not* applying function points (mainly estimates based on "intuition and experience") on small and medium large projects, and lower accuracy on large projects. The use of function points reduced the proportion of very large (>100%) effort overruns. Limitations: 1) The questionnaire data may have a low quality[2], 2) The relationship is not necessarily causal, e.g., the organizations applying estimation models may be different to other organizations. 3) Response rate not reported.
4	(Lederer and Prasad 1992), (Lederer and Prasad 1993), (Lederer and Prasad 1998), (Lederer and Prasad 2000) (reporting the same study)	Questionnaires based survey of 112 software organizations.	The algorithmic effort estimation models did not lead to higher accuracy compared with "intuition, guessing, and personal memory". Limitations: 1) The questionnaire data may have a low quality, 2) The relationship is not necessarily causal, e.g., the organizations applying estimation models may be different to other organizations. 3) Response rate of only 29%, i.e., potential biases due to differences between the organizations that answered and those that did not.
5	(Mukhopadhyay, Vicinanza et al. 1992)	Experimental comparison of the estimation accuracy of 1 expert with that of estimation models (case-based reasoning model based on previous estimation strategy of the expert, function points, and COCOMO) on 5 finished software projects.	The expert's estimates were the most accurate, but not much better than the case-based reasoning estimation model. The algorithmic estimation models (COCOMO and function points) were the least accurate. Limitations: 1) The experimental setting, 2) The algorithmic estimation models were not calibrated to the organization, 3) Only one expert.
6	(Atkinson and Shepperd 1994)	Experimental comparison of the estimation accuracy of experts (students?) with that of estimation models (analogy and function points) on 21 finished projects.	One of the analogy-based estimation models provided the most accurate estimates, then the expert judgments, then the two other analogy based models, and finally, the function point based estimation model. Limitations: 1) The experimental setting, 2) Missing information about the expert estimators and the models[3].
7	(Pengelly 1995)	Experimental comparison of the estimation accuracy of experts (activity-based estimates) with that of estimation models (Doty, COCOMO, function point, and Putnam SLIM) on 1 finished project.	The expert estimates were the most accurate. Limitations: 1) The experimental setting, 2) The estimation models were not calibrated to the organization, 3) Only one project was estimated.
8	(Jørgensen 1997)	Observation of 26 industrial projects, where 5 applied the	The function point based estimates were more accurate, mainly due to avoidance of very

[2] We include this comment on both studies applying questionnaires, because questionnaire studies typically have limited control over the quality of their data, see (Jørgensen 1995).
[3] We were only able to locate a preliminary version of this paper (from one of the authors). It is possible that the final version provides more information about the expert estimation process.

		function point estimation model, and 21 were based on expert estimates (bottom-up-based estimates).	large effort overruns. Limitations: 1) Most projects applying the function point model did also provide a bottom-up expert judgment-based effort estimate and combined these two estimates, 2) The relationship is not necessarily causal, e.g., the projects applying an estimation model may be different from the other projects.
9	(Niessink and van Vliet 1997)	Observations of 140 change tasks of an industrial software system. Comparison of the original expert estimates with estimates from formal estimation models (function points and analogy).	The analogy based-model had the most accurate estimates. The expert estimates were more accurate than the function point estimates. Limitations: 1) The expert estimates could impact the actual effort, the formal models could not, 2) The formal models used the whole data set as learning set (expect the task to be estimated), the expert estimates had only the previous tasks.
10	(Ohlsson, Wohlin et al. 1998)	Observation of 14 student software projects developing the same software.	The projects applying data from the experience database had no more accurate estimates than those which did not use the experience database. Estimation models based on previous projects with same requirement specification (analogy-based models) did not improve the accuracy. Limitations: 1) The competence level of the estimators (students), 2) The artificial context of student projects, e.g., not real customer.
11	(Walkerden and Jeffery 1999)	Experimental comparison of the estimation accuracy of 25 students with that of estimation models (analogy and regression based models) on 19 projects.	The experts' estimates had the same accuracy as the best analogy based model and better than the regression-based and the other analogy-based models. Estimates based on expert selected analogies, with a linear size adjustment, provided the most accurate effort estimates. Limitations: 1) The experimental setting, 2) The competence level of the estimators (students), 3) The project information was tailored to the estimation models, e.g., no requirement specification was available.
12	(Myrtveit and Stensrud 1999)	Experimental comparison of the estimation accuracy of 68 software professionals with that of a combination of expert estimates and models (analogy and regression), and models alone on 48 COTS projects (each participant estimated 1 project).	The models had the same or better accuracy than the combination of model and expert, and better accuracy than the unaided expert. Limitations: 1) The experimental setting, 2) The project information was tailored to the estimation models, e.g., no requirement specification was available.
13	(Bowden, Hargreaves et al. 2000)	Experimental comparison of students' ability to find "objects" as input to an estimation model in comparison with an expert system.	There was no difference in performance. Limitations: 1) The experimental setting, 2) The competence level of the estimators (students), 3) Study of input to effort estimation models, not effort estimation.
14	(Jørgensen and Sjøberg 2002b)	Observation of experts' ability to predict uncertainty of effort usage (risk of unexpected software maintenance problems) in comparison with a simple regression-based estimation model. Study based on interviews with 54 software maintainer before start and after completion of maintenance tasks.	The simple regression model predicted maintenance problems better than software maintainers with long experience. Limitations: 1) Assessment of effort estimation uncertainty, not effort estimation.
15	(Kitchenham, Pfleeger et al. 2002)	Observations of 145 maintenance tasks in a software development organization. Comparison of expert estimates with estimates based on the average of two estimation methods, e.g., the average of an expert estimates and a formal model-based estimate. The actual projects estimates were also compared with the estimates from estimation models (variants of a regression + function point-based model) based on the observed maintenance tasks.	There was no difference in estimation accuracy between the average-combined and the purely expert-based estimates. The expert estimates were more accurate than the model-based estimates. Limitations: 1) The relationship is not necessarily causal, e.g., the project combining estimation methods may be more complex than the other projects. 2) The expert estimates could impact the actual effort, the formal models could not[4].

[4] The authors conclude that the estimates did not impact the actual effort.

The results of the studies in Table 2 are not conclusive. Of the fifteen studies, we categorize five to be in favour of expert estimation (Studies 1, 2, 5, 7, and 15), five to find no difference (Studies 3, 4, 10, 11, and 13), and five to be in favour of model-based estimation (Studies 6, 8, 9, 12, and 14).

Interesting dimensions of the studies are realism (experiment versus observation), calibration of models (calibrated to an organization or not), and level of expertise of the estimator (students versus professionals). A division of the studies into categories based on these dimensions suggests that the design of the empirical studies has a strong impact on the result. All experiments applying estimation models not calibrated to the estimation environment (Studies 1, 2, 5 and 7) showed that the expert estimates were the most accurate. On the other hand, all experiments applying calibrated estimation models (Studies 10, 11, 12 and 13) showed a similar or better performance of the models. The higher accuracy of the experts in the first experimental situation can be explained by the estimation models' lack of inclusion of organization and domain specific knowledge[5]. The similar or better accuracy of the models in the second experimental situation can be explained by the lack of domain-specific knowledge of the experts, i.e., in Studies 10, 11 and 13 the estimators were students, and in Study 12 the estimation information seems to have been at a, for the software professional, unfamiliar format.

Three of the studies (Studies 8, 9, and 14) where the model-based estimates were calibrated, *and* both expert and model estimates were applied by software projects, i.e., the five observational studies (Studies 3, 4, 8, 9, and 14), show results in favour of model-based estimation. The remaining two studies of that category (Studies 3, and 4), report similar accuracy of the models and the experts. A possible explanation for the similar or higher accuracy of model-based estimates of the observational studies is that the real-world model-based estimates frequently were "expert adjusted model estimates", i.e., a *combination* of model and expert. The model-based estimates of Study 8, for example, seem to be of that type. A typical "expert adjusted model estimation"-process may be to present the output from the model to the experts. Then, the domain experts adjust the effort estimate according to what she/he believes is a more correct estimate. If this is the typical model-based estimation process, then the reported findings indicate that a combination of estimation model and expert judgment is better than pure expert estimates. More studies are needed to examine this possibility.

The above fifteen studies are not conclusive, other than that there is no substantial evidence in favour of either model or expert-based estimates. In particular, we believe that there is a need for comparative studies including a description of the actual estimation models and actual expert estimation processes in real software effort estimation situations.

None of the studies in Table 2 were designed for the purpose of examining *when* we can expect expert estimation to have the same or better estimation accuracy compared with estimation models. This is however the main question. Clearly, there exist situations were the use of formal estimation models leads to more accurate estimates, and situations where expert estimation results in higher accuracy, e.g., the two types of experimental situations described earlier. To increase the understanding of when we can expect expert estimates to have an acceptable accuracy in comparison with formal estimation models, we have tried to derive major findings from relevant human judgment studies, e.g., time estimation studies, and describe the consistence between these findings and the software-related results. This turned out to be a difficult task, and the summary of the studies described in Table 3 should be interpreted carefully, e.g., some of the findings are rather vaguely formulated, and other researchers may interpret the results from the same studies differently.

[5] There is an on-going discussion on the importance of calibrating an estimation model to a *specific* organization. While the majority of the empirical software studies, e.g., (Cuelenaere, Genuchten et al. 1987; Marouane and Mili 1989; Jeffery and Low 1990; Marwane and Mili 1991; Murali and Sankar 1997; Jeffery, Ruhe et al. 2000) report that calibration of estimation models to a specific organization led to more accurate estimates, the results in (Briand, El Emam et al. 1999; Briand, Langley et al. 2000) suggest that use of multi-organizational software development project data were just as accurate. However, the results in (Briand, El Emam et al. 1999; Briand, Langley et al. 2000) do not report from studies calibrating *general* estimation products. For example, the difference between the projects on which the original COCOMO model was developed (Boehm 1981) and projects conducted in the 1990s may be much larger than the difference between multi-organizational and organization specific project data. The evidence in favour of calibration of general estimation models in order to increase the estimation accuracy is, therefore, strong.

Table 3: Expert versus Model Estimates

Findings	**Strength of Evidence**	**Sources of Evidence**	**Consistence Between the Findings and the Results Described in Software Studies?**
Expert estimates are more accurate than model estimates when the experts possess (and efficiently apply) important domain knowledge not included in the estimation models. Model estimates are more accurate when the experts do *not* possess (or efficiently apply) important domain knowledge not included in the estimation models.	Strong	These findings are supported by "common sense", e.g., it is obvious that there exists important case-specific domain knowledge about software developers and projects that cannot be included in a general estimation model. The finding is also supported by a number of studies (mainly business forecasting studies) on the importance of specific domain knowledge in comparison with models, see (Lawrence and O'Connor 1996; Webby and O'Connor 1996; Johnson 1998; Mendes, Counsell et al. 2001) for reviews on this topic. However, as pointed out by Dawes (1986), based on studies of clinical and business judgment, the correspondence between domain knowledge and estimation skills is easily over-rated. Meehl (1957) summarizes about 20 studies comparing clinical judgment with judgment based on statistical models. He found that the models had the same or better performance in all cases. The same negative result was reported by Dawes (1986). The results in favour of models seems to be less robust when the object to be estimated include human behavior, e.g., traffic safety (Hammond, Hamm et al. 1987).	Yes. All studies where the models were *not* calibrated to the organizational context and the estimators had domain knowledge (Studies 1, 2, 5 and 7) report that the expert estimates were more accurate. All studies were the estimators had little relevant domain knowledge (due to the lack of requirement specification, lack of experience or project information tailored to the estimation models), *and* the estimation models were calibrated to the organizational context (Studies 10, 11, 12 and 13) report that the models had the same or better performance.
Expert estimates are more accurate than model estimates when the uncertainty is low. Model estimates are more accurate when the uncertainty is high, e.g., when the project is much larger than previous projects.	Medium	The majority of studies (mainly business forecasting studies) support this finding, e.g., (Braun and Yaniv 1992; Shanteau 1992; O'Connor, Remus et al. 1993; Hoch and Schkade 1996; Soll 1996). However, a few studies suggest that uncertain situations favour expert judgment, e.g., the study described in (Sanders and Ritzman 1991) on business related time series forecasting.	Mixed. Study 3 reports that high uncertainty did *not* favour the use of (function point-based) estimation model. Similarly, Study 9 reports results suggesting that low uncertainty (homogeneous tasks) did not favour expert estimates compared with an analogy-based model. An investigation of the available studies on this topic suggests that high uncertainty favour the estimation models *only if* the uncertainty is included in the estimation model. If, however, a new software task is uncertain because it represents a new type of situation not included in model's learning data set, e.g., reflects the development of a project much larger than the earlier projects, then the models are likely to be less accurate. Similar results on how uncertainty impact the expert estimation performance are reported in (Goodwin and Wright 1990) on time series forecasting.

Experts use simple estimation strategies (heuristics) and perform just as well or better than estimation models when these simple estimation strategies (heuristics) are valid. Otherwise, the strategies may lead to biased estimates.	Strong	The results reported in (Josephs and Hahn 1995; Todd and Benbasat 2000), describing studies on time planning and general decision tasks, indicate that the estimation strategies used by unaided experts were simple, even when the level of expert knowledge was high. Increasing the time pressure on the estimators may lead the experts to switch to even simpler estimation strategies, as reported in the business forecasting study described in (Ordonez and Benson III 1997). Gigerenzer and Todd (1999) present a set of human judgment studies, from several domains, that demonstrate an amazingly high accuracy of simple estimation strategies (heuristics). Kahneman et al. (1982), on the other hand studied similar judgment tasks and found that simple strategies easily led to biased estimates because the heuristics were applied incorrectly, i.e., they demonstrated that there are situations where the simple estimation strategies applied by experts are not valid. Unfortunately, it may be difficult to decide in advance whether a simple estimation strategy is valid or not.	Yes. The software development estimation experiment reported in (Jørgensen and Sjøberg 2001b) suggests that the experts applied the so-called "representativeness heuristic", i.e., the strategy of finding the most similar previous projects without regarding properties of other, less similar, projects (see also discussion in Section 4.5). Most of the estimators applied a valid version of this, but some of them interpreted representativeness too "narrow", which lead to biased estimates. Similarly, Study 14 suggests that the low performance in assessing estimation uncertainty of experienced software maintainers were caused by misuse of the "representativeness heuristic".
Experts can be strongly biased and misled by irrelevant information, e.g., towards over-optimism. Estimation models are less biased.	Strong	Substantial evidence supports this finding, e.g., (Kahneman, Slovic et al. 1982; Blattberg and Hoch 1990; Lim and O'Connor 1996; Connolly and Dean 1997; Makridakis, Wheelwright et al. 1998, p. 500-501; Whitecotton, Sanders et al. 1998; Hill, Thomas et al. 2000) reporting results from various domains. In particular relevant are the studies on the "planning fallacy" (Kahneman and Tversky 1979), i.e., the studies on people's tendency to provide too optimistic prediction of own performance in spite of knowledge about their previous over-optimism. Buehler et al. (1997) summarize studies on possible cognitive and motivational reasons for the planning fallacy.	Yes. The studies that describe expert and model estimates actually used by industrial software projects and report the size of the individual projects' effort over-runs (Studies 3 and 8) suggest that the risk of large effort over-runs was reduced when applying estimation models. The software development estimation results described in (Jørgensen and Sjøberg 2001a) suggest that an early estimate based on little information strongly biased the re-estimation, although the estimators were told not to use the early estimate as input, i.e., irrelevant information strongly misled the estimators.

An interesting observation is that the software development expert estimates are not systematically worse than the model-based estimates, such as the expert estimates in most other studied professions. For example, Dawes (1986) reports that the evidence against clinical expert judgment, compared with formal models, is overwhelming. Many of the studies described in Table 2, on the other hand, suggest that software development experts have the same or better accuracy as the formal estimation models. We believe that the two most important reasons for this difference in results are:

- The importance of specific domain knowledge (case-specific data) is higher in software development projects than in most other studied human judgment domains. For example, while most clinical diseases are based on stable biological processes with few, well-established diagnostic indicators, the relevant indicators of software development effort may be numerous, their relevance unstable and not well-established. For example, Wolverton (1974) found that: "*There is a general tendency on the part of designers to gold-plate their individual parts of any system, but in the case of software the tendency is both stronger and more difficult to control than in the case of hardware*." How much a particular project member tend to gold-plate, i.e., to improve the quality beyond what is expected by the customer, is hardly part of any estimation model, but can be known by an experienced project leader. According to Hammond et al. (1987) a "fit" between the type of estimation (human judgment) task and the selected estimation approach is essential, i.e., if a task is an expert estimation (intuition) inducing task, then the experts provide the most accurate estimates and when the task is a model estimation (analysis) inducing task then the models provided the most accurate estimates. As we interpret Hammond et al., many software development effort estimation tasks are expert estimation inducing tasks.
- The performance of the software development estimation models is poorer than estimation models in most other studied human judgment domains. For example, although there has been much research on the shape of the software "production function", i.e., relation between input and output parameters, for several years, no agreement has been reached. Dolado (2001), for example, investigated the relationship between software size and effort on 12 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.

On the other hand, we do *not* believe that the software development experts are more skilled estimators than experts in other domains. On the contrary, as reported in (Jørgensen and Sjøberg 2001a; Jørgensen and Sjøberg 2002b) the focus on learning estimation skills from software development experience seems to be very low.

Many of the shortcomings of expert estimation may be reduced when following well-documented estimation principles. In the following sections we present and discuss 12 expert estimation principles that have improvement of expert estimation as goal.

4 Reduce Situational and Human Biases

Lederer et al. (1990) describe a "rational" and a "political" model of the estimation process, based on interviews with 17 software managers. The rational model describes the estimation process as in most text-books on estimation, i.e., as a rational process with estimation accuracy as the only goal, while the political model describes the estimation process more as a "tug-of-war" with individual motives, differing goals, and power conflicts. While some of the biases resulting from a "tug-of-war" are situational, e.g., the wish to get a contract, others are more inherent human, e.g., the general need for positive feedback from other people. This section suggests six estimation principles aiming at reducing the size of situational and human biases:

- Evaluate estimation accuracy, but avoid high evaluation pressure.
- Avoid conflicting estimation goals.
- Ask the estimators to justify and criticize their estimates.
- Avoid irrelevant and unreliable estimation information.
- Use documented data from previous development tasks.
- Find estimation experts with highly relevant domain background and good estimation records.

A general framework for identifying and handling the situational and human biases is described in (Meyer and Booker 1991, p. 44-53).

4.1 Evaluate Estimation Accuracy, but Avoid High Evaluation Pressure

Several human judgment studies suggest that a high motivation for accuracy, for example when people feel personally responsible, perceive that the estimation task is very important or receive monetary rewards for accurate estimates, actually *decreases* the estimation accuracy (Sieber 1974; Armstrong, Denniston Jr. et al. 1975; Cosier and Rose 1977). Pelham and Neter (1995) suggest that this decrease in human judgment accuracy is mainly a problem in the case of difficult judgments, whereas high motivation for accuracy increases the estimation accuracy in cases with easy judgments. Their findings are consistent with the large number of studies on the effect of "evaluation apprehension", e.g., (Sanders 1984). An increased awareness of being evaluated seems to increase the level of so-called "dominant responses" (instincts) on cost of reflective responses (Zajonc 1965), i.e., evaluation leads to more instinct and less reflection. That effect may be very robust, e.g., Zajonc et al. (1969) measured a decrease in performance by cockroaches completing a maze when other cockroaches were present. When reflections and analyses are important and the task is difficult, as in many software development estimation situations, a strong perception of evaluation may therefore lead to less accurate estimates.

These results are, at first sight, not consistent with the results reported from the empirical software development studies on this topic. For example, 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 software estimation studies (Weinberg and Schulman 1974; Jørgensen and Sjøberg 2001a) 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.

The different findings are, in our opinion, not in conflict. There is no reason to believe that software professionals are different from other estimators, i.e., an increased perception of accuracy evaluation may easily lead to decreased estimation accuracy of software projects. However, evaluations may also lead to: 1) The "self-fulfilling prophecy" effect of software effort estimates, e.g., that an over-optimistic initial estimate and a high focus on estimation accuracy lead to actions that make that estimate more realistic as reported in the software project simulation study (Abdel-Hamid, Sengupta et al. 1999), and 2) An increase in "self-critical thinking" as in the study of first-job salary and exam results prediction of students reported in (Shepperd, Fernandez et al. 1996). For example, when the accountability is high people may be motivated to spend more time and collect more relevant information to achieve an accurate estimate. The total effect of accuracy evaluation, therefore, depends on the strength of the pressure due to the accuracy evaluation, the flexibility of the work (determining the possible effect from the "self-fulfilling prophecy"), and the increased degree of "self-critical thinking" as a consequence of the evaluation. Software managers should focus on achieving the benefits from accuracy evaluation, while avoiding the disadvantages. In our opinion, this means that the estimation accuracy should be part of the projects' evaluation criteria, but that a strong pressure from accuracy accountability or reward/punishment should be avoided. In addition, means to ensure "self-critical thinking" should be introduced, e.g., through estimation checklists and described estimation processes.

4.2 Avoid Conflicting Goals

There are conflicting estimation goal in situations where the estimation process is impacted by other goals (evaluations) than the accuracy of the estimate. This section focuses on two important instances of conflicting estimation goals: 1) The conflicts between "bid", "planned effort" and "most likely effort", and 2) The conflict between "wishful thinking" and "realism".

Jørgensen and Sjøberg (2001a) report that, frequently, there was no distinction between "bid", "planned effort" and "most likely effort" when estimating software development effort. Similar results, i.e., that the distinction between planning and estimation are "blurred", are reported in the time-estimation studies described in (Edwards and Moores 1994; Goodwin 1998). The decisions on "bid", "planned effort" and "most likely effort", however, have conflicting goals. 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. The conflict between these goals, together with the lack of separation of them, may hinder realism of the expert estimates. We have not found any software studies on the impact of this conflict on accuracy of effort estimate. However, applying common sense and the results described in the human judgment studies (Cosier and Rose 1977; Keen 1981; Buehler, Griffin et al. 1997), where conflicting goals were reported to reduce the realism of the estimates, we believe that the evidence against mixing the goals of "bid", "planned effort" and "most likely effort" are fairly strong.

The results from many human judgment studies indicate that people get over-optimistic when predicting own performance, i.e., they have problems separating "wish" and "realism". A summary of these studies is described by Harvey (2001). Potential reasons for this over-optimism, or "planning fallacy" (Kahneman and Tversky 1979), are the "I am above average"-bias (Klein and Kunda 1994), and the lack of distinction between "best case" and "most realistic case" (Newby-Clark, Ross et al. 2000). A general phenomenon seems to be that the level of over-optimism increases with the level of control (Koehler and Harvey 1997), e.g., a software developer responsible for the whole task to be estimated is supposed to be more over-optimistic than a project leader that plans and supervises the work of other project members. This over-optimism may be difficult to reduce, and in (Newby-Clark, Ross et al. 2000) it was found that the only effective method was to let someone other than the executing person predict the work. The same conclusion is reported in (Harvey 2001): *"someone other than the person(s) responsible for developing and implementing a plan of action should estimate its probability of success.*" Buehler et al. (1994) found that the cause of an increased realism, when estimating other peoples work, was the increase in use of previous experience, i.e., while estimating own work induces mental work on how to complete the task (construction), estimating other people's work induces reflections on how much effort similar tasks required (history reflections). Unfortunately, we have not been able to find any published software development estimation specific study on the topic of estimating own work or other people's work[6].

Similarly to the discussion in Section 4.1, there are advantages of estimating own work. For example, if there is a high level of flexibility in how to implement a software specification, then an initially over-optimistic estimate of own work may lead to actions that make the estimate more realistic. The decision whether to estimating own work or not may therefore be a trade-off between the potential advantages, e.g., higher motivation for low use of effort, and the disadvantages, e.g., the strong tendency of over-optimism. In situations where there are small opportunities for "self-fulfilling prophecies", e.g., when the flexibility of the project work is strongly limited, then the software developers should, optimally, *not* estimate their own work. In real projects, however, estimation of own work may be the only option, e.g., because there are no other experts on a particular task. In such cases, it is especially important to be aware of the typical over-optimism and apply the de-biasing estimation principles described in this paper.

An illustrative example of a conflict between wishful thinking and realism when predicting own performance is described in (Griffin and Buehler 1999): "*Canadians expecting an income-tax refund were asked to predict when they would complete and mail in their tax forms. These respondents had indicated that they typically completed this chore about 2 weeks before the due rate; however, when asked about the current year, they predicted that they would finish, on average, about 1 month in advance of the due date. In fact, only 30% of the respondents were finished by their predicted data - on average they finished, as usual, about 2 weeks before the deadline.*"

There are other, obviously unfortunate, variants of the conflict between "wishful thinking" and "realism", e.g., the "software estimation game" described in (Thomsett 1996): "*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 both puts an unfortunate pressure on the estimator and leads to conflicting goals, i.e., a conflict between "be realistic" and "please the manager".

Software professionals should learn to identify estimation goals different from accuracy, and try to avoid or at least reduce the impact from them. In particular, software professionals should learn to identify when a person has a particularly strong interest in the outcome, e.g., when a person strongly want the project to be started. In this kind of conflicting goals situation, the highly involved person cannot be expected to provide realistic estimates, even when she/he is the person with the longest and most relevant experience.

4.3 Ask Estimators to Justify and Criticize Their Estimates.

Expert estimation of effort is frequently a "constructive" process. The estimators try to imagine how to build the software, which pieces that are necessary to develop and the effort needed to implement and integrate the pieces. Empirical results from human judgment studies suggests that this type of process easily lead the estimator into the mode of "confirming theories on how to complete the project", rather than "reject incorrect hypotheses and assumptions" (Brehmer 1980; Koehler 1991). This means that the estimators' *confidence* in their estimates depend more on the amount of effort they spent working on it, than on the actual accuracy of the estimate. Justification and critique of own estimates may have several important advantages related to this problem. It may:

- increase of the accuracy of the estimate, particularly in high uncertainty situations (Hagafors and Brehmer 1983),
- lead to a more analytical estimation process and reduce the risk of using too simple estimation strategies (Hammond 1996),
- improve the level of confidence in the estimate (Koriat, Lichtenstein et al. 1980), and
- improve the compensation for missing information (Brenner, Koehler et al. 1996).

All the above studies were general human judgment studies, e.g., studies based on real-world clinical judgment tasks, business tasks, or estimates of so-called "almanac quantities". We have found no published software development estimation study on this topic.

However, as part of an experiment conducted by the author of this paper, we asked thirteen software professionals to estimate the effort they would need to implement a specified timeshift-swapping system for hospital nurses. When the effort estimates were completed, the estimators were asked to list reasons why their estimate could be wrong, i.e., a critique of their own estimates. The average number of reasons listed were 4.3, ranging from 2 to 8. Finally, the estimators were asked to consider a change of their original estimates in light of their critique. Nine out of the thirteen software professionals increased their estimates of most likely effort, four

[6] In a recent, unpublished, study of sixty small and medium large software development tasks, we find supporting evidence for this difference between estimation own and other peoples work. The difference in level of over-optimism was significant, but not very large.

of them more than 25%. The average increase in effort estimate was, however, only 10%, and four of the participants actually decreased their estimates. We had no opportunity to let the software professionals develop the software, i.e., we had no information about the realism of their estimates. However, the small, on average, adjustments suggested by our results mean that, although potentially helpful to improve estimation realism, we should not expect that justification and criticism improve the realism of estimates very much. If the initial estimate is hugely over-optimistic, a justification and critique may only improve the realism to some extent. A possible reason for this limited impact is described in (Einhorn and Hogarth 1978), based on studies on clinical judgment and probability assessments. Estimators are typically not very skilled in searching for weakening information when evaluating their own estimates.

In spite of the expected small impact on the realism of the estimate, we believe that justification and criticism are sound and low-cost elements of improvements of expert estimates.

4.4 Avoid Irrelevant and Unreliable Estimation Information

It is easy to accept that irrelevant and unreliable information should be avoided. However, we have yet to see a checklist or estimation process effectively implementing this estimation principle. This may reflect the belief that expert estimators are able to filter out irrelevant and unreliable information when facing it. There are, however, several human judgment studies that suggest that this is not always the case, and that expert estimates may be strongly impacted by irrelevant information, even when the estimators know that the information is irrelevant. For example:

- Whitecotton et al. (1998) report that people are just as good as models to provide financial forecasts when presented with the same highly relevant information, but less accurate when irrelevant information is included.
- Lim and O'Connor (1996) report from business related time series predictions that an adjustment of an estimate for new information was not sufficient when the initial estimate was highly inaccurate, i.e., that the unreliable initial estimate strongly impacted the subsequent estimates. The software development estimation study described by Abdel-Hamid et al.(1993) confirm this result.
- Tversky and Kahneman (1974) report, based on general knowledge tasks, that the estimators were impacted by irrelevant information, because it was included in the question, i.e., people may have an implicit tendency to regard information as important when it is presented in the same context as the estimation problem.
- Ettenson et al. (1987) report that domain experts (financial auditing) were better than novices to focus on the most relevant information, i.e., the experts applied less information compared with the novices. Selection of proper experts may, therefore, be important to avoid strong impact from irrelevant information.
- Jørgensen and Sjøberg (2002a) report that the information about the software development cost expected by the customer had a strong impact on the estimate even when the estimators were told that the customer knew nothing about the realistic costs and that the information should be regarded as irrelevant for the estimation task. More surprisingly, this impact from the customer expectation was strongly underestimated by the software professionals.

Consequently, it is may not be sufficient to warn against irrelevant information or instruct people to consider information as unreliable. The only safe approach seems to *avoid* irrelevant and unreliable information. For example, it may be difficult to provide realistic effort estimates if the customer expects an unrealistically low level of cost, and the estimator knows this. Then, the only safe option may be to find a new estimator, without that knowledge.

4.5 Use Documented Data from Previous Development Tasks

Use of documented data means that that the expert estimators have the opportunity to apply a more analytic estimation strategy and consequently, be less prone to human and situational biases. Benefits from use of documented software project data are reported by Lederer and Prasad (1992), who found that software project cost overruns were associated with lack of documented data from previous tasks, i.e., high reliance on "personal memory". Without documented data people seem to both over-react to immediate past information, as reported in the time series prediction study (Remus, O'Connor et al. 1995), and rely too much on the "representativeness" estimation strategy, see the software development estimation study (Jørgensen and Sjøberg 2002b). The "representativeness" estimation strategy means, for example, that people use the actual effort of the *most* similar (most representative) recalled task as staring point for the estimate without regarding the distribution of effort of other similar tasks. This strategy works well when the most similar task is sufficiently similar, represents the typical use of effort on such tasks, and the estimation uncertainty is low. The strategy may, however, lead to inaccurate estimates when the need for adjustment is large, as illustrated in the business forecasting study (Blattberg and Hoch 1990), or the expected impact from the "regression toward the mean"[7] is high, as reported

[7] The impact from "regression toward the mean" is based on the observation that high or low performance tends to be

in the human judgment and software estimation studies (Kahneman and Tversky 1973; Nisbett and Ross 1980; Jørgensen 2002).

A similar argument for the importance of documented data is reported in the time usage estimation study (Kahneman and Lovallo 1993). That study claims that people tend to adopt an "internal" or "inside" perspective on the estimation task, when relying on their own memory, instead of documented data. This "inside" perspective leads to a concentration on case-specific planning and a neglect of "background" information, such as the distribution of completion times for similar projects or the robustness of the construction plan. An "inside" perspective may work well when the estimator has strongly relevant task experience and the situation does not induce biases, but may otherwise lead to a high degree of estimation inaccuracy. The results described in (Kahneman and Lovallo 1993) may explain the reduction of high effort overruns from use of models reported in the software development estimation studies (Heemstra and Kusters 1991; Jørgensen 1997). The use of estimation models increases the use of historical data and, consequently, removes the potentially large biases from expert estimators' "inside view" and the use of the "representativeness" estimation strategy.

The software development estimation results reported in (Walkerden and Jeffery 1999) indicate that a semi-automated use of documented data leads to the best estimation accuracy. They found, similar to the business forecasting results reported by Blattberg and Hoch (1990), that people were good at finding analogies, but did not adjust properly for large differences between the task to be estimated and the most similar tasks. A semi-automated process of using people to find the relevant analogues and a simple formula for adjustments for differences had the best estimation accuracy. If the need for adjustments is large, simple models supporting the adjustments seem to be especially important.

Overall, we believe that the potential benefits from use of documented data are similar to the potential benefits from use of estimation models, i.e., avoidance of very inaccurate estimates and reduction of human biases.

4.6 Find Experts with Relevant Domain Background and Good Estimation Records

Recently we conducted an estimation survey of the estimation processes of eighteen experienced software project leaders. Included in that survey was a question about how the project leaders selected experts to provide the effort estimates. While all the project leaders described that they emphasized domain and development experience, only four of them described that they applied information about the peoples' previous estimation accuracy, and only two that they tried to get information about the estimation process applied by the estimator. An underlying assumption of the selection of estimation experts was, as we interpreted it, that "the people most competent in solving the task should estimate it". While this assumption can be true, see (Sanders and Ritzman 2001) for an overview of supporting expert judgment studies from various domains, we believe that the following refinements of the assumption are important:

- The relevance of experience is sometimes very "narrow", i.e., only applicable in very similar situations, see (Skitmore, Stradling et al. 1994; Ericsson and Lehmann 1996) for overviews from different domains.
- Jørgensen and Sjøberg (2002b) report that software maintainers with application specific experience had fewer maintenance problems, but did *not* predict their own work more accurately. Similarly, Lichtenstein and Fischhoff (1977) report that the level of over-optimism when estimating the quality of their own answers on "general knowledge" questions was independent of the actual correctness of the answers, i.e., the level of expertise. These findings conflict those reported in statistical forecasting studies, e.g., (Sanders and Ritzman 2001). An examination of the studies suggests that the explanation is the difference between involved and uninvolved estimators. While all the results described in (Sanders and Ritzman 2001) are derived from studies where the estimators were uninvolved observers, the results described in (Lichtenstein and Fischhoff 1977; Jørgensen and Sjøberg 2002b) are from studies where own work was estimated. A large benefit from domain experience on estimation accuracy may, consequently, require that the estimator is an uninvolved observer.
- Klayman and Gonzalez-Vallejo (1999) report, based on tasks from several domains, that people get over-confident in the accuracy of their estimates when receiving a set of estimation tasks more difficult than what they usually get.
- Stone and Opel (2000) report that having estimation expertise is not the same as being skilled in knowing the uncertainty of an estimate. Their experiment, based on art history related judgment tasks, suggest that these two types of expertise require different types of feedback and training.

Consequently, we cannot safely assume that people knowing much about a task are good at estimating it, nor can we assume that people good at estimating are good at knowing how uncertain their estimates are. For this reason, there should be separate records on these three characteristics (know-how, know-how-much, and know-how-uncertain) for each individual. Knowing much about a task may, for example, be useful for the

followed by more average performance, in particular when the variance (uncertainty) is high. This means, for example, that when the most similar task had an unusual high performance and the estimation uncertainty is high, then we should estimate effort closer to the average performance than the effort value of the most similar task (Jørgensen 2002).

development of the work breakdown structure. People with good estimation records should be consulted when estimating the most likely effort. People good at estimating uncertainty should be consulted when assessing the uncertainty of the estimate. These three skills are different and may require different estimators, training, and feedback, see Section 6.

5 Support the Estimation Process

There are many ways of supporting the experts' estimation processes. This section provides and discusses the expert estimation principles:

- Estimate both top-down and bottom-up, independently of each other
- Use estimation checklists
- Combine estimates from different sources
- Assess the uncertainty of the estimate

5.1 Estimate Both Top-Down and Bottom-Up, Independently of Each Other

There are different strategies of decomposing the estimation problem, e.g., phase-based decomposition, functionality-based decomposition, additive, multiplicative, or combinations of these types. Most studies support the, on average, improvement from decomposing an estimation problem, see for example the multi-domain survey on this topic in (MacGregor 2001). There are, however, studies that indicate no benefits of decomposition. For example, Connolly and Dean (1997) found that the estimation accuracy improved from software task decomposition in only one out of two experiments. Vicinanza et al. (1991) found that the expert applying a top-down (analogy)-based software development estimation process was more accurate than the experts relying on a decomposition-based process. Moløkken (2002) found that the software professionals applying a bottom-up software development estimation process were more over-optimistic than those applying a more top-down estimation process. Similarly, no benefits were found from applying the function point software development estimation model "bottom-up", instead of the common "top-down" application (Yau and Gan 1995). It is common sense that some tasks are too complex to understand and estimate as a whole, i.e., that decomposition is necessary to understand some problems. The results from the software estimation studies, however, suggest that there are potential problems with decomposing the software development estimation problem applying the "bottom-up" (additive decomposition) that are avoided through a top-down estimation process.

We suggest that a bottom-up estimation process, e.g., estimation of the activities described in a work breakdown structure (Tausworthe 1980), should be combined with a top-down estimation process, e.g., the process of estimating the project as a whole through comparison with similar completed projects. We believe that these two estimation processes should be conducted independently of each other, to avoid the "anchoring effect"[8], i.e., that one estimate gets strongly impacted by the other as reported in the software development effort study (Jørgensen and Sjøberg 2001a). If there are large deviations between the estimates provided by the different processes, and estimation accuracy is important, then more estimation information and/or independent estimation experts should be added. Alternatively, a simple average of the two processes can be applied (more on the benefits of different strategies of combining estimates in Section 5.3). Our belief in the usefulness of this "do-both" principle is based on the complementary strengths and weaknesses of top-down and bottom-up-based expert estimates as described in Table 4.

Table 4: Top-Down versus Bottom-Up

	Top-Down (as a Whole)	**Bottom-Up (Decomposed)**
Strengths	More robust with respect to forgotten activities and unexpected events. Encourages "distributional" (history-based) thinking.	Leads to increased understanding of the execution and planning of the project (how-to knowledge).
Weaknesses	Does not lead to increased understanding of the execution and planning of the project. Depends strongly on the proper selection and availability of similar projects from memory or project documentation.	Easy to forget activities and underestimate unexpected events. Depends strongly on selection of software developers with proper experience. Does not encourage history-based criticism of the estimate and its assumptions.

[8]: *Anchoring: "the tendency of judges' estimates (or forecasts) to be influenced when they start with a 'convenient' estimate in making their forecasts. This initial estimate (or anchor) can be based on tradition, previous history or available data."* (Armstrong 2001b).

The claimed benefits and weaknesses in Table 4 are supported by results reported in, e.g., the software studies (Hill, Thomas et al. 2000; Moløkken 2002). Buehler et al. (1994) report a study where the difference between instructing people to use their past experience, instead of only focusing on how to complete a task, reduced the level of over-optimism in time estimation tasks. This result supports the importance of applying a strategy that induces distributional (history-based) thinking, e.g., top-down estimation strategies. Perhaps the most important part of top-down estimation is *not* that the project is estimated as a whole, but that it encourages the use of history. Other interesting results on impacts from decomposition strategies include:

- Decomposition is not useful for low-uncertainty estimation tasks, only for high-uncertainty, as reported in several forecasting and human judgment studies (Armstrong, Denniston Jr. et al. 1975; MacGregor 2001).
- Decomposition may "activate" too much knowledge (including non-relevant knowledge). For this reason, predefined decompositions, e.g., predefined work breakdown structures, activating only relevant knowledge should be applied. The human judgment study reported in (MacGregor and Lichtenstein 1991) supports this result.

In sum, the results suggest that bottom-up-based estimates only lead to improved estimation accuracy if the uncertainty of the whole task is high, i.e., the task is too complex to estimate as a whole, and, the decomposition structure activates relevant knowledge only. The validity of these two conditions is, typically, not possible to know in advance and applying both top-down and bottom-up estimation processes, therefore, reduces the risk of highly inaccurate estimates.

5.2 Use Estimation Checklists

The benefits of checklists are not controversial and are based on, at least, four observations:

- Experts easily forget activities and underestimate the effort required to solve unexpected events. Harvey (2001) provides an overview of forecasting and human judgment studies on how checklists support people in remembering important variables and possibilities that they would otherwise overlook.
- Expert estimates are inconsistent, i.e., the same input may result in different estimates. For example, experts seem to respond to increased uncertainty with increased inconsistency (Harvey 2001). Checklists may increase the consistency, and hence the accuracy, of the expert estimates.
- People tend to use estimation strategies that require minimal computational effort, at the expense of accuracy, as reported in the time estimation study described in (Josephs and Hahn 1995). Checklists may "push" the experts to use more accurate expert estimation strategies.
- People have a tendency to consider only the options that are presented, and underestimate the likelihood of the other options, as reported in the "fault tree" study described in (Fischhoff, Slovic et al. 1978). This means that people have a tendency to "out of sight, out of mind". Checklists may encourage the generation of more possible outcomes.

Interestingly, there is evidence that checklists can bring novices up to an expert level. For example, Getty et al. (1988) describe a study were general radiologists were brought up to the performance of specialist mammographers using a checklist.

Although we have experienced that many software organizations find checklists to be one of their most useful estimation tools, we have not been able to find any empirical study on how different types of checklists impact the accuracy of software effort estimation. Common sense and studies from other domains leave, however, little doubt that checklists are an important means to improve expert estimation. An example of a checklist (aimed at managers that review software project estimates) is provided in (Park 1996): (1) Are the objectives of the estimates clear and correct? 2) Has the task been appropriately sized? 3) Are the estimated cost and schedule consistent with demonstrated accomplishments on other projects? 4) Have the factors that affect the estimate been identified and explained? 5) Have steps been taken to ensure the integrity of the estimating process? 6) Is the organization's historical evidence capable of supporting a reliable estimate? 7) Has the situation changed since the estimate was prepared? This type of checklist clearly supports the estimation reviewer to remember important issues, increases the consistency of the review process, and "pushes" the reviewer to apply an appropriate review process.

A potential "by-product" of a checklist is the use of it as a simple means to document previous estimation experience. The aggregation of the previous estimation experience into a checklist may be easier to use and have more impact on the estimation accuracy compared with a large software development experience databases containing project reports and estimation data (Jørgensen, Sjøberg et al. 1998).

5.3 Obtain and Combine Estimates from Different Experts and Approaches

When two or more experts provide estimates of the same task, the optimal approach would be to use only the most accurate estimates. The individuals' estimation accuracies are, however, not known in advance and a combination of several estimates has been shown to be superior to selecting only one of the available estimates.

See (Clemen 1989) for an extensive overview of empirical studies from various domains on this topic. The two software studies we were able to find on this topic are consistent with the findings from other domains. These studies report an increase in estimation accuracy through averaging of the individual estimate (Höst and Wohlin 1998) and group discussions (Jørgensen and Moløkken 2002). Based on the extensive evidence in favour of combining estimates the question should *not* be whether we should combine or not, but how?

There are many alternative combination approaches for software project estimates. A software project leader can, for example, collect estimates of the same task from different experts and then weight these estimates according to level of the experts' level of competence. Alternatively, the project leader can ask different experts to discuss their estimates and agree on an estimate. The choice of combination strategy and the benefits from combined estimates depend on a number of variables. The variables are, according to Hogarth's model (1978): 1) Number of experts, 2) The individuals' (expected) estimation accuracy, 3) The degree of biases among the experts, and 4) The inter-correlation between the experts' estimates. A human judgment study validating Hogarth's model is described in (Ashton 1986). Our discussion on combination of estimates will be based on these four variables, and, a fifth variable not included in Hogarth's model[9]: 5) The impact of combination strategy.

Number of experts (1): The number of expert estimates to be included in the combined estimate depends on their expected accuracy, biases and inter-correlation. Frequently, the use of relatively few (3-5) experts with different backgrounds seems to be sufficient to achieve most of the benefits from combining estimates, as reported in the study of financial and similar types of judgments described in (Libby and Blashfield 1978).

The accuracy and biases of the experts (2+3): A documented record of the experts' previous estimation accuracy and biases is frequently not available or not relevant for the current estimation task. However, the project leaders may have informal information indicating for example the level of over-optimism or expertise of an estimator. This information should be used, with care, to ensure that the accuracy of the experts is high and that individual biases are not systematically in one direction.

The inter-correlation between the experts (4): A low inter-correlation between the estimators is important to exploit the benefits from combining estimates. Studies reporting the importance of this variable in business forecasting and software development estimation contexts are (Armstrong 2001a; Jørgensen and Moløkken 2002). A low inter-correlation can be achieved when selecting experts with different backgrounds and roles, or experts applying different estimation processes.

Combination process (5): There are several approaches of combining expert estimates. One may take the average of individual software development effort estimates (Höst and Wohlin 1998), apply a structured software estimation group process (Taff, Borchering et al. 1991), select the expert with the best estimate on the previous task (Ringuest and Tang 1987), or apply the well-documented Delphi-process (Rowe and Wright 2001). A comprehensive overview of combination strategies is described in (Chatterjee and Chatterjee 1987). While the choice of combination strategy may be important in some situations, there are studies, e.g., the forecasting study described in (Fisher 1981), that suggest that most meaningful combination processes have similar performance. Other human judgment and forecasting studies, however, found that averaging the estimates was the best combination strategy (Clemen 1989), or that a group-based processes led to the highest accuracy (Reagan-Cirincione 1994; Henry 1995; Fischer and Harvey 1999). In (Moløkken 2002) it is reported that a group discussion-based combination of individual software development effort estimates was more accurate than the average of the individual estimates, because the group discussion led to new knowledge about the interaction between people in different roles. Similar results, on planning of R&D projects, were found in (Kernaghan and Cooke 1986; Kernaghan and Cooke 1990). This increase in knowledge through discussions is an important advantage of group-based estimation processes compared with "mechanical" combinations, such as averaging. However, the evidence in favour of group-based combinations is not strong. For example, group discussion may lead to more biased estimates (either more risky or more conservative) depending on the group processes and the individual goals, as illustrated in the financial forecasting study described in (Maines 1996).

In summary, it seems that the most important part of the estimation principle is to combine estimates from different sources (with, preferably, high accuracy and low inter-correlation), not exactly how this combination is conducted.

5.4 Assess the Uncertainty of the Estimate

Important reasons for the importance of assessing the uncertainty of an effort estimate are:

- The uncertainty of the estimate is important information in the planning of a software project (McConnel 1998).
- An assessment of the uncertainty is important for the learning from the estimate, e.g., low estimation accuracy is not necessarily an indicator of low estimation skills when the software development project work is highly uncertain (Jørgensen and Sjøberg 2002b).

[9] This is no shortcoming of Hogarth's model, since his model assumes that the combined estimate is based on the *average* of the individual estimates.

- The process of assessing uncertainty may lead to more realism in the estimation of most likely software development effort. The software estimation study reported in (Connolly and Dean 1997) supports this finding, but there are also contradictory findings, e.g., time usage estimation study described in (Newby-Clark, Ross et al. 2000).

We recommend, similarly to the forecasting principles described by Armstrong (2001d), that the uncertainty of an estimate is assessed through a prediction interval. For example, a project leader may estimate that the most likely effort of a development project is 10000 work-hours and that it is 90% certain (confidence level) that the actual use of effort will be between 5000 and 20000 work-hours. Then, the interval [5000, 20000] work-hours is the 90% prediction interval of the effort estimate of 10000 work-hours.

A confidence level of *K*% should, in the long run, result in a proportion of actual values inside the prediction interval (hit rate) of *K*%. However, Connolly and Dean (1997) report that the hit rates of students' effort predictions intervals were, on average, 60% when a 90% confidence level was required. Similarly, (Jørgensen, Teigen et al. 2002) report that the activity effort hit rates of several industrial software development projects were all less than 50%[10], i.e., the intervals were much too narrow.

This type of over-confidence seems to be found in most other domains, see for example (Alpert and Raiffa 1982; Lichtenstein, Fischhoff et al. 1982; McClelland and Bolger 1994; Wright and Ayton 1994; Bongaarts and Bulatao 2000). As reported earlier, Lichtenstein and Fischhoff (1977) report that the level of over-confidence was unaffected by differences in intelligence and expertise, i.e., we should *not* expect that the level of over-confidence is reduced with more experience. Arkes (2001) gives a recent overview of studies from different domains on over-confidence, supporting that claim. Potential reasons for this over-confidence are:

- *Poor statistical knowledge.* The statistical assumptions underlying prediction intervals and probabilities are rather complex, see for example (Christensen 1998). Even with sufficient historical data the estimators may not know how to provide, for example, a 90% prediction interval of an estimate.
- *Estimation goals in "conflict" with the estimation accuracy goal.* The software professionals' goals of appearing skilled and providing "informative" prediction intervals may be in conflict with the goal of sufficiently wide prediction intervals, see for example the human judgment studies (Yaniv and Foster 1997; Keren and Teigen 2001) and our discussion in Section 4.1.
- "*Anchoring effect".* Several studies from various domains, e.g., (Kahneman, Slovic et al. 1982; Jørgensen and Sjøberg 2002a), report that people typically provide estimates influenced by an anchor value and that they are not sufficiently aware of this influence. The estimate of the most likely effort may easily become the anchor value of the estimate of minimum and maximum effort. Consequently, the minimum and maximum effort will not be sufficiently different from the most likely effort in high uncertainty situations.
- *"Tendency to over-estimate own skills".* Kruger and Dunning (1999) found a tendency to over-estimate one's own level of skill in comparison with the skill of other people. This tendency increased with decreasing level of skill. A potential effect of the tendency is that information about previous estimation inaccuracy of similar projects has insufficient impact on a project leaders uncertainty estimate, because most project leaders believe to be more skilled than average.

In total, there is strong evidence that the traditional, unaided expert judgment-based assessments of estimation uncertainty through prediction intervals are biased toward over-confidence, i.e., too narrow prediction intervals. An uncertainty elicitation process that seems to reduce the over-confidence in software estimation contexts is described in (Jørgensen and Teigen 2002). This process, which is similar to the method proposed by (Seaver, Winterfeldt von et al. 1978), proposes a simple change of the traditional uncertainty elicitation process:

1. Estimates the most likely effort.
2. Calculate the minimum and maximum effort as fixed proportions of the most likely effort. For example, an organisation could base these proportions on the NASA-guidelines (NASA 1990) of software development project effort intervals and set the minimum effort to 50% and the maximum effort to 200% of the most likely effort.
3. Decide on the confidence level, i.e., assess the probability that the actual effort is between the minimum and maximum effort.

Steps 2 and 3 are different from the traditional uncertainty elicitation process, where the experts are instructed to provide minimum and maximum effort values for a given confidence level, e.g., a 90% confidence level. The differences may appear minor, but include a change from "self-developed" to "mechanically" developed minimum and maximum values. Minimum and maximum values provided by oneself, as in the traditional elicitation process, may be used to indicate estimation skills, e.g., to show to other people that "my estimation work is of a high quality". Mechanically calculated minimum and maximum values, on the other hand, may reduce this "ownership" of the minimum and maximum values, i.e., lead to a situation similar to when experts evaluate estimation work conducted by other people. As discussed in Section 4.2, it is much easier to be realistic when assessing other peoples performance, compared with own performance. In addition, as

[10] The industrial projects did not have a consistent use of confidence level, but, typically, let the estimators decide how to interpret minimum and maximum effort. Nevertheless, most meaningful interpretations of minimum and maximum effort should lead to higher hit rates than 40-50%.

opposed to the traditional process, there is no obvious anchor value that influences the prediction intervals toward over-confidence when assessing the appropriate confidence level of a mechanically derived prediction interval. Other possible explanations for the benefits of the proposed approach, e.g., easier learning from history, are described in (Jørgensen and Teigen 2002). The proposed approach was evaluated on the estimation of a set of maintenance tasks and found to improve the correspondence between confidence level and hit rate significantly (Jørgensen and Teigen 2002).

An alternative elicitation method, not yet evaluated in software contexts, is to ask for prediction intervals based on low confidence levels, e.g., to ask a software developer to provide a 60% instead of a 90% prediction interval. This may reduce the level of over-confidence, because, as found by (Roth 1993), people are generally better calibrated in the middle of a probability distribution than in its tails.

6 Provide Estimation Feedback and Training Opportunities

It is hard to improve estimation skills without feedback and training. Lack of estimation feedback and training may, however, be a common situation in software organizations (Hughes 1996a; Jørgensen and Sjøberg 2002b). The observed lack of feedback of software organizations means that it is no large surprise that increased experience did not lead to improved estimation accuracy in the studies (Hill, Thomas et al. 2000; Jørgensen and Sjøberg 2002b). Similarly, many studies from other domains report a lack of correlation between amount of experience and estimation skills. Hammond (1996, 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"*.

Learning estimation skills from experience can be difficult (Jørgensen and Sjøberg 2000). In addition to sufficient and properly designed estimation feedback, estimation improvements may require the provision of training opportunities (Ericsson and Lehmann 1996). This section discusses feedback and training principles for improvement of expert estimates.

6.1 Provide Feedback on Estimation Accuracy and Development Task Relations

There has been much work on frameworks for "learning from experience" in software organizations, e.g., work on experience databases (Basili, Caldierea et al. 1994; Houdek, K et al. 1998; Jørgensen, Sjøberg et al. 1998; Engelkamp, Hartkopf et al. 2000) and frameworks for Post-Mortem (project experience) reviews (Birk, Dingsøyr et al. 2002). These studies do not, as far as we know, provide empirical results on the relation between type of feedback and estimation accuracy improvement. The only software study on this topic (Ohlsson, Wohlin et al. 1998), to our knowledge, suggest that outcome feedback, i.e., feedback relating the actual outcome to the estimated outcome, did not improve the estimation accuracy. Human judgment studies from other domains support this disappointing lack of estimation improvement from outcome feedback, see for example (Balzer, Doherty et al. 1989; Benson 1992; Stone and Opel 2000). This is no large surprise, since there is little estimation accuracy improvement possible from the feedback that, for example, *"the effort estimate was 30% too low"*. One situation were outcome feedback is reported to improve the estimation accuracy is when the estimation tasks are "dependent and related" and the estimator initially was under-confident, i.e., underestimated her/his own knowledge on general knowledge tasks (Subbotin 1996). In spite of the poor improvement in estimation accuracy, outcome feedback is useful, since it improves the assessment of the uncertainty of an estimate (Stone and Opel 2000; Jørgensen and Teigen 2002). Feedback on estimation accuracy should, for that reason, be included in the estimation feedback.

To improve the estimation accuracy, several studies from various domains 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). A possible method to provide this type of feedback is the use "experience reports" or "post mortem" review processes.

When analysing the impacts from different variables on the use of effort and the estimation accuracy, i.e., the "task relation oriented feedback", it important to understand interpretation biases and the dynamics of software projects, e.g.,:

- The "hindsight bias", e.g., the tendency to interpret cause-effect relationships as more obvious after it happen than before, see (Fischhof 1975; Stahlberg, Eller et al. 1995) for general human judgement studies on this topic.
- The tendency to confirm rules and disregard conflicting evidence, as illustrated in the human judgement studies (Camerer and Johnson 1991; Sanbonmatsu, Sharon et al. 1993) and our discussion in Section 4.3.
- The tendency to apply a "deterministic" instead of a "probabilistic" learning model. For example, assume that a software project introduces a new development tool for the purpose of increasing the efficiency and that the project has many inexperienced developers. The actual project efficiency turns out to be lower than that of the previous projects and the actual effort, consequently, becomes much higher than the estimated

effort. A (naïve) deterministic interpretation of this experience would be that "new tools decrease the development efficiency if the developers are inexperienced'. A probabilistic interpretation would be to consider other possible scenarios (that did not happen, but could have happen) and to conclude that it seems to be more than 50% likely that the combination of new tools and inexperienced developers lead to a decrease in efficiency. This ability to think in probability-based terms can, according to Brehmer (1980), hardly be derived from experience alone, but must be taught. Hammond (1996) suggest that the ability to understand relationships in terms of probabilities instead of purely deterministic connections is important for correct learning in situations with high uncertainty.

- The potential impact of the estimate on the actual effort as reported in the software estimation studies (Abdel-Hamid and Madnik 1983; Jørgensen and Sjøberg 2001a), i.e., the potential presence of a"self-fulfilling prophecy". For example, software projects that over-estimate the "most likely effort" may achieve high estimation accuracy if the remaining effort is applied to improve ("gold-plate") the product.
- The potential lack of distinction between "plan" and "estimate", see discussion in Section 4.2.
- The variety of reasons for high or low estimation accuracy, as pointed out in the industrial software estimation study (Jørgensen, Moen et al. 2002). Low estimation accuracy may, for example, be the results of poor project control, high project uncertainty, low flexibility in delivered product (small opportunity to "fit" the actual use of effort to the estimated), project members with low motivation for estimation accuracy, high project priority on time-to-market, "bad luck", or, of course, poor estimation skills.
- A tendency to asymmetric cause-effect analyses dependent on high or low accuracy, i.e., high estimation accuracy is explained as good estimation skills, while low estimation accuracy is explained as impact from external uncontrollable factors. Tan and Lipe (1997) found, in a business context, 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."*

In many human judgment situations with high uncertainty and unstable task relations, there are indications on that even task relation-oriented feedback is not sufficient for learning (Schmitt, Coyle et al. 1976; Bolger and Wright 1994), i.e., the situations do simply not enable learning from experience. For this reason, it is important to recognize when there is nothing to learn from experience, as reported in the software estimation study (Jørgensen and Sjøberg 2000).

A problem with most feedback on software development effort estimates is that it takes too much time from the point-of-estimation to the point-of-feedback. This is unfortunate, since it has been shown that immediate feedback strongly improves the estimation learning and accuracy, as illustrated in the human judgment studies (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 to realistic estimates. Although it is easy to over-rate the possibility to learn from feedback, it is frequently the only realistic opportunity for learning, i.e., even if the benefits are smaller than we like to believe, software organizations should do their best to provide properly designed estimation feedback.

6.2 Provide Estimation Training Opportunities

Frequently, real software projects provide too little information to draw valid conclusions about cause-effects (Jørgensen and Sjøberg 2000). Blocher et al. (1997) report similar results based on studies of people's analytical procedures. Bloher et al. attribute the cause-effect problems to the lack of learning about what would have happened if we had not done what we did, and the high number of alternative explanation for an event. Furthermore, they argue that learning requires the development of causal models for education, training and professional guidance. The importance of causal domain models for training is supported by the human judgment results described in (Bolger and Wright 1994). Similar reasons for learning problems, based on a review of studies on differences in performance between experts and novices in many different domains, are provided by Ericsson and Lehmann (1996). They claim that it is not the amount of experience but the amount of "deliberate training" that determines the level of expertise. They interpret deliberate training as *"individualized training activities especially designed by a coach or teacher to improve specific aspects of an individual's performance through repetition and successive refinement"*. This importance of training is also supported by the review of human judgment studies described in (Camerer and Johnson 1991), suggesting 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 should 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 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 understood, 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 pure "learning from experience".

7 Conclusions and Further Research

The two main contributions of this paper are:

- A systematic review of papers on software development effort expert estimation.
- An extensive examination of relevant human judgment studies to validate expert estimation "best practice" principles.

The review concludes that expert estimation is the dominant strategy when estimating the effort of software development projects, and that there is no substantial evidence supporting the superiority of model estimates over expert estimates. There are situations where expert estimates are more likely to be more accurate, e.g., situations where experts have important domain knowledge not included in the models or situations when simple estimation strategies provide accurate estimates. Similarly, there are situations where the use of models may reduce large situational or human biases, e.g., when the estimators have a strong personal interest in the outcome. The studies on expert estimation are summarized through an empirical evaluation of the twelve principles: 1) Evaluate estimation accuracy, but avoid high evaluation pressure, 2) Avoid conflicting estimation goals, 3) Ask the estimators to justify and criticize their estimates, 4) Avoid irrelevant and unreliable estimation information, 5) Use documented data from previous development tasks, 6) Find estimation experts with relevant domain background and good estimation record, 7) Estimate top-down and bottom-up, independently of each other, 8) Use estimation checklists, 9) Combine estimates from different experts and estimation strategies, 10) Assess the uncertainty of the estimate, 11) Provide feedback on estimation accuracy and task relations, 12) Provide estimation training opportunities. We find that there is evidence supporting all these principles and, consequently, that software organizations should apply them.

The estimation principles are to some extent based on results from other domains than software development, or represent only one type of software projects and experts. For this reason there is a strong need for better insight into the validity and generality of many of the discussed topics. In particular we plan to continue with research on:

- When to use expert estimation and when to use estimation models.
- How to reduce the over-optimism bias when estimating own work applying expert estimation.
- How to select and combine a set of expert estimates.
- The benefits of "deliberate" estimation training.

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

References

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

Abdel-Hamid, T. K., K. Sengupta and D. Ronan 1993. Software project control: An experimental investigation of judgment with fallible information. *IEEE Transactions on Software Engineering* 19(6): 603-612.

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

Alpert, M. and H. Raiffa 1982. A progress report on the training of probability assessors. *Judgment under uncertainty: heuristics and biases*. Ed. A. Tversky. Cambridge, Cambridge University Press**:** 294-305.

Arkes, H. R. 2001. Overconfidence in judgmental forecasting. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 495-515.

Armstrong, J. S. 2001a. Combining forecasts. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 417-440.

Armstrong, J. S. 2001b. The forecasting dictionary. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 761-824.

Armstrong, J. S. 2001c. Selecting forecasting methods. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 365-386.

Armstrong, J. S. 2001d. Standards and practices for forecasting. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 679-732.

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

Ashton, R. H. 1986. Combining the judgments of experts: How many and which ones? *Organizational Behaviour and Human Decision Processes* 38(3): 405-414.

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

Ayton, A. 1998. How bad is human judgment? *Forecasting with judgment*. Ed. G. Wright and P. Goodwin. New York, Johh Wiley & Son**:** 237-268.

Balzer, W. K., M. E. Doherty and R. J. O'Connor 1989. Effects of cognitive feedback on performance. *Psychological Bulletin* 106(3): 410-433.

Basili, V., H. Caldierea and D. Rombach 1994. The Experience Factory. *Encyclopedia of Software Engineering*. Ed. 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.

Betteridge, R. 1992. Successful experience of using function points to estimate project costs early in the life-cycle. *Information and Software Technology* 34(10): 655-658.

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

Blattberg, R. C. and S. J. Hoch 1990. Database models and managerial intuition: 50% model + 50% manager. *Management Science* 36: 887-899.

Blocher, E., M. J. Bouwman and C. E. Daves 1997. Learning from experience in performing analytical procedures. *Training Research Journal* 3: 59-79.

Boehm, B., C. Abts and S. Chulani 2000. Software development cost estimation approaches - A survey. *Annals of software engineering* 10: 177-205.

Boehm, B. and K. Sullivan 1999. Software economics: Status and prospects. *Information and Software Technology* 41: 937-946.

Boehm, B. W. 1981. *Software engineering economics*. New Jersey, Prentice-Hall.

Boehm, B. W. 1984. Software engineering economics. *IEEE Transactions on Software Engineering* 10(1): 4-21.

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

Bongaarts, J. and R. A. Bulatao 2000. *Beyond Six Billion: Forecasting the World's Population*, National Academy Press.

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

Braun, P. A. and I. Yaniv 1992. A case study of expert judgment: economists' probabilities versus base-rate model forecasts. *Journal of Behavioral Decision Making* 5(3): 217-231.

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

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

Briand, L. C., K. El Emam, D. Surmann, I. Wieczorek and K. D. Maxwell 1999. An assessment and comparison of common software cost estimation modeling techniques. *International Conference on Software Engineering*, Los Angeles, USA, ACM, New York: 313-323.

Briand, L. C., T. Langley and I. Wieczorek 2000. A replicated assessment and comparison of common software cost modeling techniques. *International Conference on Software Engineering*, Limerick, Ireland, ACM, New York: 377-386.

Briand, L. C. and I. Wieczorek 2002. Resource estimation in software engineering. *Encyclopedia of software engineering*. Ed. J. J. Marcinak. New York, John Wiley & Sons.

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

Buehler, R., D. Griffin and M. Ross 1994. Exploring the "Planning fallacy": Why people underestimate their task completion times. *Journal of Personality and Social Psychology* 67(3): 366-381.

Camerer, C. F. and E. J. Johnson 1991. The process-performance paradox in expert judgment: How can experts know so much and predict so badly? *Towards a general theory of expertise*. Ed. K. A. Ericsson and J. Smith, Cambridge University Press**:** 195-217.

Chatterjee, S. and S. Chatterjee 1987. On combining expert opinion. *American Journal of mathematical and management sciences* 7(3&4): 271-295.

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

Chulani, S., B. Boehm and B. Steece 1999. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering* 25(4): 573-583.

Clemen, R. T. 1989. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting* 5(4): 559-583.

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

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

Cuelenaere, A. M. E., M. J. I. M. Genuchten and F. J. Heemstra 1987. Calibrating a software cost estimation model: Why and how. *Information and Software Technology* 29(10): 558-567.

Dawes, R. M. 1986. Proper and improper linear models. *International Journal of Forecasting* 2: 5-14.

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

Edwards, J. S. and T. T. Moores 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.

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

Engelkamp, S., S. Hartkopf and P. Brössler 2000. Project Experience Database: A Report Based on First Practical Experience. *PROFES*, Oulu, Finland, Springer-Verlag: 204-215.

Ericsson, K. A. and A. C. Lehmann 1996. Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual Review of Psychology* 47: 273-305.

Ettenson, R., J. Shanteau and J. Krogstad 1987. Expert judgment: Is more information better. *Psychological reports* 60(1): 227-238.

Fairley, R. E. 1992. Recent advantages in software estimation techniques. *International Conference on Software Engineering*, Melbourne, Australia: 382-391.

Fischer, I. and N. Harvey 1999. Combining forecasts: What information do judges need to outperform the simple average. *International Journal of Forecasting* 15(3): 227-246.

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.

Fischhoff, B., P. Slovic and S. Lichtenstein 1978. Fault trees: Sensitivity of estimated failure probabilities to problem representation. *Journal of Experimental Psychology: Human Perception and Performance* 4(2): 330-334.

Fisher, G. W. 1981. When oracles fail--a comparison of four procedures for aggregating subjective probability forecasts. *Organizational Behaviour and Human Performance* 28(1): 96-110.

Getty, D. J., R. M. Pickett, S. J. D'Orsi and J. A. Swets 1988. Enhanced interpretation of diagnostic images. *Investigative radiology* 23: 244-252.

Gigerenzer, G. and P. M. Todd 1999. *Simple heuristics that make us smart*. New York, Oxford University Press.

Goodman, P. A. 1992. Application of cost-estimation techniques: Industrial perspective. *Information and Software Technology* 34(6): 379-382.

Goodwin, P. 1998. Enhancing judgmental sales forecasting: The role of laboratory research. *Forecasting with judgment*. Ed. G. Wright and P. Goodwin. New York, John Wiley & Sons**:** 91-112.

Goodwin, P. and G. Wright 1990. Improving judgmental time series forecasting: A review of the guidance provided by research. *International Journal of Forecasting* 9: 147-161.

Griffin, D. and R. Buehler 1999. Frequency, probability, and prediction: Easy solutions to cognitive illusions? *Cognitive Psychology* 38(1): 48-78.

Hagafors, R. and B. Brehmer 1983. Does having to justify one's judgments change nature of the judgment

process? *Organizational Behaviour and Human Decision Processes* 31(2): 223-232.
Hammond, K. R. 1996. *Human judgement and social policy: Irreducible uncertainty, inevitable error, unavoidable injustice.* New York, Oxford University Press.
Hammond, K. R., R. M. Hamm, J. Grassia and T. Pearson 1987. Direct comparison of the efficacy of intuitive and analytical cognition in expert judgment. *IEEE Transactions on systems, man, and cybernetics* 17(5): 753-770.
Harvey, N. 2001. Improving judgment in forecasting. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 59-80.
Heemstra, F. J. 1992. Software cost estimation. *Information and Software Technology* 34(10): 627-639.
Heemstra, F. J. and R. J. Kusters 1991. Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems* 1(4): 223-237.
Henry, R. A. 1995. Improving group judgment accuracy: Information sharing and determining the best member. *Organizational Behaviour and Human Decision Processes* 62: 190-197.
Hihn, J. and H. Habib-Agahi 1991a. 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.
Hihn, J. and H. Habib-Agahi 1991b. Cost estimation of software intensive projects: A survey of current practices. *International Conference on Software Engineering*: 276-287.
Hill, J., L. C. Thomas and D. E. Allen 2000. Experts' estimates of task durations in software development projects. *International Journal of Project Management* 18(1): 13-21.
Hoch, S. J. and D. A. Schkade 1996. A psychological approach to decision support systems. *Management Science* 42(1): 51-64.
Hogarth, R. M. 1978. A note on aggregating opinions. *Organizational Behaviour and Human Performance* 21(1): 40-46.
Houdek, F., S. K and W. E 1998. Establishing Experience Factories at Daimler-Benz An Experience Report. *International Conference on Software Engineering*, Kyoto, Japan: 443-447.
Hughes, R. T. 1996a. Expert judgement as an estimating method. *Information and Software Technology* 38(2): 67-75.
Hughes, R. T. 1996b. Expert judgement as an estimation method. *Information and Software Technology* 38: 67-75.
Höst, M. and C. Wohlin 1997. A subjective effort estimation experiment. *Information and Software Technology* 39(11): 755-762.
Höst, M. and C. Wohlin 1998. An experimental study of individual subjective effort estimations and combinations of the estimates. *International Conference on Software Engineering*, Kyoto, Japan, IEEE Comput. Soc, Los Alamitos, CA, USA: 332-339.
Jeffery, D. R. and G. Low 1990. Calibrating estimation tools for software development. *Software Engineering Journal* 5(4): 215-221.
Jeffery, D. R., M. Ruhe and I. Wieczorek 2000. A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data. *Information and Software Technology* 42(14): 1009-1016.
Johnson, E. J. 1998. Expertise and decision under uncertainty: Performance and process. *The nature of expertise*. Ed. M. T. H. Chi, R. Glaser and M. J. Farr. Hillsdale, N. J., Lawrence Erlbaum**:** 209-228.
Josephs, R. and E. D. Hahn 1995. Bias and accuracy in estimates of task duration. *Organizational Behaviour and Human Decision Processes* 61(2): 202-213.
Jørgensen, M. 1995. The quality of questionnaire based software maintenance studies. *ACM SIGSOFT - Software Engineering Notes* 20(1): 71-73.
Jørgensen, M. 1997. An empirical evaluation of the MkII FPA estimation model. *Norwegian Informatics Conference*, Voss, Norway, Tapir, Oslo: 7-18.
Jørgensen, M. 2002. Software Effort Estimation by Analogy and "Regression Toward the Mean". *To appear in: Journal of Systems and Software*.
Jørgensen, M., L. Moen and N. Løvstad 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 K. Moløkken 2002. Combination of software development effort prediction intervals: Why, when and how? *Fourteenth IEEE Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, Ischia, Italy.
Jørgensen, M. and D. Sjøberg 2000. The importance of not learning from experience. *European Software Process Improvement 2000 (EuroSPI'2000)*, Copenhagen: 2.2-2.8.
Jørgensen, M. and D. I. K. Sjøberg 2001a. Impact of effort estimates on software project work. *Information and Software Technology* 43(15): 939-948.
Jørgensen, M. and D. I. K. Sjøberg 2001b. Software process improvement and human judgement heuristics.

Scandinavian Journal of Information Systems 13: 99-121.

Jørgensen, M. and D. I. K. Sjøberg 2002a. The Impact of Customer Expectation on Software Development Effort Estimates. *Submitted to International Journal of Project Management*.

Jørgensen, M. and D. I. K. Sjøberg 2002b. Impact of experience on maintenance skills. *Journal of Software Maintenance and Evolution: Research and practice* 14(2): 123-146.

Jørgensen, M., D. I. K. Sjøberg and R. Conradi 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. and K. H. Teigen 2002. Uncertainty Intervals versus Interval Uncertainty: An Alternative Method for Eliciting Effort Prediction Intervals in Software Development Projects. *Proceedings of: International conference on Project Management (ProMAC)*, Singapore, 343-352.

Jørgensen, M., K. H. Teigen and K. Moløkken 2002. Better sure than safe? Overconfidence in judgment based software development effort prediction intervals. *Submitted to Journal of Systems and Software*.

Kahneman, D. and D. Lovallo 1993. Timid choices and bold forecasts: A cognitive perspective on risk taking. *Management Science* 39(1): 17-31.

Kahneman, D., P. Slovic and A. Tversky 1982. *Judgment under uncertainty: Heuristics and biases*. Cambridge, United Kingdom, Cambridge University Press.

Kahneman, D. and A. Tversky 1973. On the psychology of prediction. *Psychological Review* 80(4): 237-251.

Kahneman, D. and A. Tversky 1979. Intuitive predictions: Biases and corrective procedures. *TIMS Studies in Management Science* 12: 313-327.

Keen, P. G. W. 1981. Information systems and organizational change. *Social Impacts of Computing* 24(1): 24-33.

Keren, G. and K. H. Teigen 2001. Why is p=.90 better than p=.70? preference for definitive predictions by lay consumers of probability judgments. *Psychonomic Bulletin & Reviews* 8(2): 191-202.

Kernaghan, J. A. and R. A. Cooke 1986. The contribution of the group process to successful project planning in R&D settings. *IEEE Transactions on Engineering Management* 33(3): 134-140.

Kernaghan, J. A. and R. A. Cooke 1990. Teamwork in planning innovative projects: Improving group performance by rational and interpersonal interventions in group process. *IEEE Transactions on Engineering Management* 37(2): 109-116.

Kitchenham, B., S. L. Pfleeger, B. McColl and S. Eagan 2002. A case study of maintenance estimation accuracy. *To appear in: Journal of Systems and Software*.

Klayman, J., J. B. Soll, V. C. Gonzalez and S. Barlas 1999. Overconfidence: It depends on how, what and whom you ask. *Organizational Behaviour and Human Decision Processes* 79(3): 216-247.

Klein, W. M. and Z. Kunda 1994. Exaggerated self-assessments and the preference for controllable risks. *Organizational behavior and human decision processes.* 59(3): 410-427.

Koehler, D. J. 1991. Explanation, imagination, and confidence in judgment. *Psychological bulletin* 110(3): 499-519.

Koehler, D. J. and N. Harvey 1997. Confidence judgments by actors and observers. *Journal of Behavioral Decision Making* 10(3): 221-242.

Koriat, A., S. Lichtenstein and B. Fischhoff 1980. Reasons for confidence. *Journal of Experimental Psychology: Human Learning and Memory* 6(2): 107-118.

Kruger, J. and D. Dunning 1999. Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology* 77(6): 1121-1134.

Kusters, R. J. 1990. Are software cost-estimation models accurate? *Information and software technology* 32: 187-190.

Kusters, R. J., M. J. I. M. Genuchten and F. J. Heemstra 1990. Are software cost-estimation models accurate? *Information and Software Technology* 32(3): 187-190.

Lawrence, M. and M. O'Connor 1996. Judgement or models: The importance of task differences. *Omega, International Journal of Management Science* 24(3): 245-254.

Lederer, A. L., R. Mirani, B. S. Neo, C. Pollard, J. Prasad and K. Ramamurthy 1990. Information system cost estimating: a management perspective. *MIS Quarterly* 14(2): 159-176.

Lederer, A. L. and J. Prasad 1992. Nine management guidelines for better cost estimating. *Communications of the ACM* 35(2): 51-59.

Lederer, A. L. and J. Prasad 1993. Information systems software cost estimating: a current assessment. *Journal of Information Technology* 8(1): 22-33.

Lederer, A. L. and J. Prasad 1998. A causal model for software cost estimating error. *IEEE Transactions on Software Engineering* 24(2): 137-148.

Lederer, A. L. and J. Prasad 2000. Software management and cost estimating error. *Journal of Systems and Software* 50(1): 33-42.

Libby, R. and R. K. Blashfield 1978. Performance of a composite as a function of the number of judges. *Organizational Behaviour and Human Performance* 21(2): 121-129.

Lichtenstein, S. and B. Fischhoff 1977. Do those who know more also know more about how much they know? *Organizational Behaviour and Human Decision Processes.* 20(2): 159-183.

Lichtenstein, S., B. Fischhoff and L. D. Phillips 1982. Calibration of probabilities: The state of the art to 1980. *Judgment under uncertainty: Heuristics and biases*. Ed. A. Tversky. Cambridge, Cambridge University Press.

Lim, J. S. and M. O'Connor 1996. Judgmental forecasting with time series and causal information. *International Journal of Forecasting* 12(1): 139-153.

Londeix, B. 1995. Deploying realistic estimation (field situation analysis). *Information and Software Technology* 37(12): 655-670.

MacGregor, D. G. 2001. Decomposition for judgmental forecasting and estimation. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 107-123.

MacGregor, D. G. and S. Lichtenstein 1991. Problem structuring aids for quantitative estimation. *Journal of Behavioral Decision Making* 4(2): 101-116.

Maines, L. A. 1996. An experimental examination of subjective forecast combination. *International Journal of Forecasting* 12(2): 223-233.

Makridakis, S., S. C. Wheelwright and R. J. Hyndman 1998. *Forecasting : methods and applications*. New York, John Wiley & Son.

Marouane, R. and A. Mili 1989. Economics of software project management in Tunisia: basic TUCOMO. *Information and Software Technology* 31(5): 251-257.

Marwane, R. and A. Mili 1991. Building tailor-made software cost model: Intermediate TUCOMO. *Information and Software Technology* 33(3): 232-238.

McClelland, A. G. R. and F. Bolger 1994. The calibration of subjective probabilities: theories and models 1980-94. *Subjective probability*. Ed. P. Ayton. Chichester, John Wiley.

McConnel, S. 1998. *Software project survival guide*, Microsoft Press.

Meehl, P. E. 1957. When shall we use our heads instead of the formula? *Journal of Counseling Psychology* 4(4): 268-273.

Mendes, E., S. Counsell and N. Mosley 2001. *Measurement and effort prediction for Web applications*, Springer-Verlag, Berlin, Germany.

Meyer, M. A. and J. M. Booker 1991. *Eliciting and analyzing expert judgment: A practical guide*. Philadelphia, Pennsylvania, SIAM.

Mizuno, O., T. Kikuno, K. Inagaki, Y. Takagi and K. Sakamoto 2000. Statistical analysis of deviation of actual cost from estimated cost using actual project data. *Information and Software Technology* 42: 465-473.

Mohanty, S. N. 1981. Software cost estimation: Present and future. *Software - Practice and Experience* 11(2): 103-121.

Moløkken, K. 2002. Expert estimation of Web-development effort: Individual biases and group processes (Master Thesis). *Department of Informatics*, University of Oslo.

Mukhopadhyay, T., S. S. Vicinanza and M. J. Prietula 1992. Examining the feasibility of a case-based reasoning model for software effort estimation. *MIS Quarterly* 16(2): 155-171.

Murali, C. S. and C. S. Sankar 1997. Issues in estimating real-time data communications software projects. *Information and Software Technology* 39(6): 399-402.

Myrtveit, I. and E. Stensrud 1999. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering* 25: 510-525.

NASA 1990. *Manager's handbook for software development*. Goddard Space Flight Center, Greenbelt, MD, NASA Software Engineering Laboratory.

Newby-Clark, I. R., M. Ross, R. Buehler, D. J. Koehler and D. Griffin 2000. People focus on optimistic scenarios and disregard pessimistic scenarios when predicting task completion times. *Journal of Experimental Psychology: Applied* 6(3): 171-182.

Niessink, F. and H. van Vliet 1997. Predicting maintenance effort with function points. *International conference on software maintenance*, Bari, Italy, IEEE Comput. Soc, Los Alamitos, CA, USA: 32 - 39.

Nisbett, R. E. and L. Ross 1980. *Human inference: Strategies and shortcomings of social judgment*, Englewood Cliffs, NJ: Prentice-Hall.

O'Connor, M., W. Remus and K. Griggs 1993. Judgmental forecasting in times of change. *International Journal of Forecasting* 9(2): 163-172.

Ohlsson, M. C., C. Wohlin and B. Regnell 1998. A project effort estimation study. *Information and Software Technology* 40(14): 831-839.

Ordonez, L. and L. Benson III 1997. Decisions under time pressure: How time constraint affects risky decision making. *Organizational Behaviour and Human Decision Processes* 71(2): 121-140.

Park, R. E. 1996. A manager's checklist for validating software cost and schedule estimates. *American Programmer* 9(6): 30-35.

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.
Pelham, B. W. and E. Neter 1995. The effect of motivation of judgment depends on the difficulty of the judgment. *Journal of Personality and Social Psychology* 68(4): 581-594.
Pengelly, A. 1995. Performance of effort estimating techniques in current development environments. *Software Engineering Journal* 10(5): 162-170.
Reagan-Cirincione, P. 1994. Improving the accuracy of group judgment: A process intervention combining group facilitation, social judgment analysis, and information technology. *Organizational Behaviour and Human Decision Processes* 58(2): 246-270.
Remus, W., M. O'Connor and K. Griggs 1995. Does reliable information improve the accuracy of judgmental forecasts? *International Journal of Forecasting* 11(2): 285-293.
Ringuest, J. L. and K. Tang 1987. Simple rules for combining forecasts: Some empirical results. *Socio-Econ. Plann. Sci.* 21(4): 239-243.
Roth, P. L. 1993. Research trends in judgment and their implications for the Schmidt-Hunter global estimation procedure. *Organizational Behaviour and Human Decision Processes* 54(2): 299-319.
Rowe, G. and G. Wright 2001. Expert opinions in forecasting: The role of the Delphi process. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 125-144.
Sanbonmatsu, D. M., A. A. Sharon and E. Biggs 1993. Overestimating causality: Attributional effects of confirmatory processing. *Journal of Personality and Social Psychology* 65(5): 892-903.
Sanders, D. E. and L. P. Ritzman 1991. On knowing when to switch from quantitative to judgemental forecasts. *International Journal of Forecasting* 11(6): 27 - 37.
Sanders, G. S. 1984. Self-presentation and drive in social facilitation. *Journal of Experimental Social Psychology* 20(4): 312-322.
Sanders, N. R. and L. P. Ritzman 2001. Judgmental adjustment of statistical forecasts. *Principles of forecasting: A handbook for researchers and practitioners*. Ed. J. S. Armstrong. Boston, Kluwer Academic Publishers**:** 405-416.
Schmitt, N., B. W. Coyle and L. King 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., D. Winterfeldt von and W. Edwards 1978. Eliciting subjective probability distributions on continuous variables. *Organizational Behaviour and Human Decision Processes.* 21(3): 379-391.
Shanteau, J. 1992. Competence in experts: The role of task characteristics. *Organizational Behaviour and Human Decision Processes* 53(2): 252-266.
Shepperd, J. A., J. K. Fernandez and J. A. Quellette 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.
Simon, H. A. 1987. Making management decisions: The role of intuition and emotion. *Acad. Management Exec.* 1: 57-63.
Skitmore, R. M., S. G. Stradling and A. P. Tuohy 1994. Human effects in early stage construction contract price forecasting. *IEEE Transactions on Engineering Management* 41(1): 29-40.
Soll, J. B. 1996. Determinants of overconfidence and miscalibration: The roles of random error and ecological structure. *Organizational Behaviour and Human Decision Processes* 65(2): 117-137.
Stahlberg, D., F. Eller, A. Maass and D. Frey 1995. We knew it all along: Hindsight bias in groups. *Organizational Behaviour and Human Decision Processes* 63(1): 46-58.
Stone, E., R and R. B. Opel 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.
Taff, L. M., J. W. Borchering and J. W. R. Hudgins 1991. Estimeetings: development estimates and a front-end process for a large project. *IEEE Transactions on Software Engineering* 17(8): 839-849.
Tan, H.-T. and M. G. Lipe 1997. Outcome effects: the impact of decision process and outcome controllability. *Journal of Behavioral Decision Making* 10(4): 315-325.
Tausworthe, R. C. 1980. The work breakdown structure in software project management. *Journal of Systems and Software* 1(3): 181-186.
Thomsett, R. 1996. Double Dummy Spit and other estimating games. *American Programmer* 9(6): 16-22.
Todd, P. and I. Benbasat 2000. Inducing compensatory information processing through decision aids that facilitate effort reduction: An experimental assessment. *Journal of Behavioral Decision Making* 13(1): 91-106.
Tversky, A. and D. Kahneman 1974. Judgment under uncertainty: Heuristics and biases. *Science* 185: 1124-

1131.
van Genuchten, M. and H. Koolen 1991. On the use of software cost models. *Information and Management* 21: 37-44.
Verner, J. M., S. P. Overmyer and K. W. McCain 1999. In the 25 years sins The Mythical Man-Month what have we learned about project management? *Information and Software Technology* 41: 1021-1026.
Vicinanza, S. S., T. Mukhopadhyay and M. J. Prietula 1991. Software effort estimation: An exploratory study of expert performance. *Information systems research* 2(4): 243-262.
Walkerden, F. and D. R. Jeffery 1997. Software cost estimation: A review of models, process, and practice. *Advances in Computers* 44: 59-125.
Walkerden, F. and R. Jeffery 1999. An empirical study of analogy-based software effort estimation. *Journal of Empirical Software Engineering* 4(2): 135-158.
Webby, R. G. and M. J. O'Connor 1996. Judgemental and Statistical Time Series Forecasting: A Review of the Literature. *International Journal of Forecasting* 12(1): 91-118.
Weinberg, G. M. and E. L. Schulman 1974. Goals and performance in computer programming. *Human Factors* 16(1): 70 - 77.
Whitecotton, S. M., D. E. Sanders and K. B. Norris 1998. Improving predictive accuracy with a combination of human intuition and mechanical decision aids. *Organizational Behaviour and Human Decision Processes* 76(3): 325-348.
Winklhofer, H., A. Diamantopoulos and S. F. Witt 1996. Forecasting practice: a review of the empirical literature and an agenda for future research. *International Journal of Forecasting* 12(2): 193-221.
Wolverton, R. W. 1974. The cost of developing large-scale software. *IEEE Transactions on Software Engineering* C-23(6): 615-636.
Wright, G. and P. Ayton 1994. *Subjective probability*. West Sussex, England, John Wiley.
Yaniv, I. and D. P. Foster 1997. Precision and accuracy of judgmental estimation. *Journal of behavioral decision making* 10: 21-32.
Yau, C. and L.-Y. Gan 1995. Comparing the top-down and bottom-up approaches of function point analysis: A case study. *Software Quality Journal* 4(3): 175-187.
Zajonc, R. B. 1965. Social facilitation. *Science* 149(Whole No. 3681): 269-274.
Zajonc, R. B., A. Heingarner and E. M. Herman 1969. Social enhancement and impairment of performance in the cockroach. *Journal of Personality and Social Psychology* 13(2): 83-92.