Better Sure than Safe? Overconfidence in Judgment Based Software Development Effort Prediction Intervals

Magne Jørgensen[i], Karl Halvor Teigen[ii] , Kjetil Moløkken[iii]

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

ii) University of Oslo, Department of Psychology, P.O.Box 1094 Blindern, 0317 Oslo, Norway, k.h.teigen@psykologi.uio.no

iii) Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway, kjetilmo@ifi.uio.no

Abstract

The uncertainty of a software development effort estimate can be indicated through a prediction interval, i.e., the estimated minimum and maximum effort corresponding to a specific confidence level. For example, a project manager may be "90% confident" or believe that is it "very likely" that the effort required to complete a project will be between 8,000 and 12,000 work-hours. This paper describes results from four studies (Studies A-D) on human judgement (expert) based prediction intervals of software development effort. Study A examines the accuracy of the prediction intervals in real software projects. The results suggest that the prediction intervals were generally much too narrow to reflect the chosen level of confidence, i.e., that there was a strong over-confidence. Studies B, C and D try to understand the reasons for the observed over-confidence. Study B examines the possibility that the over-confidence is related to type of experience or estimation process. Study C examines the possibility that the concept of confidence level is difficult to interpret for software estimators. Finally, Study D examines the possibility that there are unfortunate feedback mechanisms that reward over-confidence.

1 Introduction

The uncertainty of effort estimates is of considerable importance, both from a project management and a "learning from experience" point of view. For example, the project budget allocated to resolution of possible unexpected problems should depend on the levels of such uncertainty (McConnel 1998). Knowledge about the uncertainty of effort estimates is also necessary to make sense of the deviation between the estimated and actual effort (Jørgensen and Sjøberg 2002a). Large deviations between the estimated and actual effort do not necessarily indicate poor estimation skills. It could alternatively reflect high inherent uncertainty in use of effort due to the application of complex technology and development of innovative software solutions. Proper learning from experience, therefore, requires knowledge about the degree of uncertainty of the estimate.

The uncertainty of an effort estimate may be described through an effort prediction interval (PI), i.e., a minimum - maximum effort interval corresponding to a stated confidence level. For example, a project manager may estimate that the most likely effort of a project is 10,000 work-hours, and add that it is 90% certain that the actual effort will turn out to be between 8,000 and 12,000 work-hours. Here, the interval [8 000, 12 000] work-hours is the 90% PI of the effort estimate. Typically, confidence levels of 90% or higher are recommended for project planning models, see for example the PERT (Project Evaluation and Review Technique) approach described by Moder et al. (1995).

However, as far as we have observed, software project managers typically use more informal descriptions of the confidence levels, e.g., confidence levels described in terms of "highly probable", "very likely", "most pessimistic and most optimistic", etc. The exact semantic of these uncertainty statements is not clear and the interpretations may, therefore, differ strongly between different software developers. An unclear semantic means also that the accuracy of the effort PIs are difficult to evaluate and that it is difficult to use the effort PIs as input for project planning. It would, for this reason, be useful if software developers were able to provide accurate effort PIs with a well-defined semantic, i.e., a probability based confidence level.

It appears to be customary for industrial software development projects to base their effort PIs on human (expert) judgements, i.e., they have no model support to assess the uncertainty of the effort estimates. Considering the obvious importance of effort uncertainty assessments for software projects and the common use of expert judgements for this purpose, the limited research on human judgement based software development effort PIs is surprising. While there have been several studies of human judgement based software development effort estimation (expert estimates), e.g., Höst & Wohlin (1998), Jørgensen & Sjøberg (1997; 2001), and Myrtveit & Stensrud (1999), studies of risk analysis in connection with software development effort estimation, e.g., Fewster & Mendes (2001) and Känsälä (1997), we found only one study evaluating human judgement based software development effort PIs (Connolly and Dean 1997). That study reports that the actual effort used by student programmers to solve programming tasks only in 60% of the cases fell inside their 98% confidence effort PIs, i.e., the effort PIs were much too narrow to reflect a high level of certainty. Explicit attention to, and training in, establishing good minimum and maximum effort values did increase the proportion inside the PI to about 70%, which was still far from the required 98%. In other words, the effort PIs provided by software developers may be rather inaccurate.

Indications on low ability to assess development effort uncertainty were also found by Jørgensen & Sjøberg (2002a). A very simple prediction model, based only on the task size category (small vs. large task), predicted the occurrence of major unexpected problems during a maintenance task better than the maintainers did themselves. This is surprising, considering the software maintainers' extensive application and development experience. However, there are several human judgment studies from other domains, see (Dawes 1988) for an overview, that report simple linear models to be superior to the experts' "holistic" judgements, i.e., intuitive judgements based on experts' total experience. Dawes (1988) attributes experts' poor performance partly to the difficulty people have in addressing two or more aspects of a situation simultaneously. This may be particularly difficult when the integration requires knowledge about the distribution of the variables involved, e.g., uncertainty distributions of developer productivity. In other words, the disappointing results regarding software professionals' ability to assess development effort uncertainty corresponds well with other research results on this topic.

An alternative solution to human judgment based effort PIs is the use of formal prediction models, e.g., the software development uncertainty prediction models described in (Angelis and Stamelos 2000; Jørgensen and Sjøberg 2002b). Formal model based effort PIs seem to be more accurate than human judgment based, but they are also more prone to use the available uncertainty information inefficiently, i.e., they sometimes provide meaninglessly wide effort PIs (Jørgensen and Sjøberg 2002b). The presence of uncertainty information which is difficult to integrate in formal effort PI models, e.g., detailed knowledge about the project members, means that we will not be able to totally replace human judgment on the issue of effort PIs. A related problem is that there are, to our knowledge, few studies that address which factors may have an impact on the uncertainty of effort estimates.

The results reported in (Gray, MacDonnell et al. 1999) suggest that over-estimation was connected with changes on small modules and development of screens, while under-estimation was connected with changes on large modules and development of reports. The results reported in (Jørgensen and Moløkken, 2003) suggest that large software development tasks were typically under-estimated and small tasks over-estimated, that tasks where time-of-delivery was paramount had lower estimation accuracy than those with a focus on quality or cost, and that estimating own work led to more accurate estimates compared to estimating other peoples' work. There is, however, a need for much more work on this topic before we, eventually, may be able to develop good models of effort estimation uncertainty.

Earlier studies on uncertainty assessments of software projects have important limitations. In (Connolly and Dean 1997) only small student programming tasks that required less than one week of work were studied, and in (Jørgensen and Sjøberg 2002a) risk levels, not effort PIs were examined. The possibility, therefore, remains that effort PIs of more realistically sized software projects are better than these studies seem to imply.

In this article we present observational studies and experiments designed to elucidate different aspects of judgement based effort PIs. The goal is both to assess the quality of the performance and to increase our understanding of the determinants for high or low performance. Specifically, we ask how well programmers with different levels and type of expertise (students and professionals) are able to estimate the uncertainty involved in software development projects. We also investigate the meanings they attach to different levels of confidence, and their preferences regarding prediction intervals and confidence levels.

The remaining part of the paper is organised as follows: In Section 2 we describe and discuss the measures used to evaluate and analyse the performance of the effort PIs. Section 3 presents the motivations, designs and results of the individual studies. In Section 4 we discuss the results of the presented studies in light of related studies. Finally, Section 5 concludes and describes plans for further work.

2 Evaluation Measures

Effort PIs can be described and evaluated through measures of hit rate, interval width, correspondence between interval width and estimation accuracy, and the distribution of actual performance relative to the effort PIs.

Hit rate: Whereas the accuracy of individual effort estimates can be assessed by comparing them to actual efforts, individual effort PIs have no obvious corresponding actual values. In the long run, however, a *K*% confidence level should correspond to hit rate of *K*%, so for instance a 90% probability for a predicted effort interval requires 90% of the actual effort values to fall within the upper and lower bound of the estimated interval (a 90% hit rate). A mismatch between the confidence level and the hit rate implies that the effort PIs are inaccurate. If the hit rate is lower than the confidence level, we observe overconfidence. We measure the hit rate as:

$$HitRate = \frac{1}{n}\sum_i h_i, \quad h_i = \begin{cases} 1, & \min_i \le Act_i \le \max_i \\ 0, & Act_i > \max_i \vee Act_i < \min_i \end{cases},$$

where $\min_i$ and $\max_i$ are, respectively, the minimum and maximum values of the PI for the effort estimate of task *i*, Act_i is the actual effort of task *i* and *n* is the number of estimated tasks.

Example: Assume that a software developer provides effort PIs on 100 software development tasks. Only 68 of them include the actual effort. The HitRate is then 68%.

Relative width: Of two sets of PIs with the same (or similar) hit rates, the set with the narrower intervals is more informative, and also indicative of a higher level of expertise or more efficient use of the uncertainty information than the wider intervals. For example, a person who is only guessing may end up with an adequate hit rate if 90% of his 90% PIs are extremely wide. To compare PIs for tasks of different magnitudes, the *relative* width of an effort PI may be a better indicator than absolute PI widths. We define the relative width of an effort PI as:

$PIWidth_i = (Max_i - Min_i) / Est_i$, where Est_i is the estimated (most likely) effort of task *i*.

Example: Assume that the software developers A and B provide effort PIs on 100 software development tasks with similar HitRate, e.g., the HitRate of A is 68% and HitRate of B is 70%. Software developer A has, due to his better estimation skills and ability to collect relevant information, on average provided 50% narrower effort PIs than B, who provided very wide effort PIs to reflect his lack of estimation skills and knowledge. Although the software developers A and B have similar HitRate, the effort PIs of software developer A would clearly be more useful as input to the planning of the software development tasks, due to lower PIWidths.

Width-Accuracy Correlations: If the effort PIs are based on extensive knowledge about the underlying uncertainty of each individual activity and different activities have different levels of uncertainty, we expect a positive correlation between PIWidth and the accuracy of the estimated most likely effort, i.e., narrow PIs should go together with accurate individual estimates, whereas wide PIs can be associated with inaccurate estimates. If, on the other hand, the effort PIs are based on the uncertainty level of most available software projects, on “wild guesses”, or the projects have very similar underlying uncertainty, we will only expect weak correlations.

We use the Balanced Relative Error (BRE) in our width-accuracy correlation analyses. BRE is defined as:

$BRE_i = (Act_i - Est_i) / min(Act_i, Est_i)$, where $min(Act_i, Est_i)$ is the lowest value of Act_i and Est_i.

We use BRE as an accuracy measure instead of the more common accuracy measure MRE (=|Act-Est|/Act), see (Conte, Dunsmore et al. 1986). BRE allows, for example, a three times too high and a three times too low effort estimate to yield the same absolute accuracy value (abs(BRE)=2). This is not the case with MRE. No matter how much the actual effort is underestimated, MRE cannot exceed 1, whereas overestimations lead to MRE-values with no upper limits. For this reason, we believe that BRE is a more appropriate measure than MRE for analyses of linear relationships (i.e., correlations) especially in data sets including projects with strongly underestimated effort. Further discussions on the properties of BRE and MRE can be found in Jørgensen and Sjøberg (2002b) and Miyazaki, Takanou et al. (1991; 1994).

Example: Assume that there are two organizations, A and B. In organisation A, there was a high ($r = 0.8$) and significant ($p = 0.03$) correlation between estimation accuracy and interval width. Assume that a project leader in organisation A asks for an estimate of the most likely effort and the effort PI of a project sub-task. If the effort PI is narrow, then the project leader can, correctly, assume that the estimate is fairly accurate. In organisation B, on the other hand, there was no correlation between estimation accuracy and interval width. In that organisation, a project leader cannot assume that a narrow interval indicates an estimate with high accuracy.

Uncertainty distribution correspondence: The effort PIs studied in this paper do not describe the shape of the effort probability distributions, for instance whether the probabilities of different levels of effort usage follow a triangular, rectangular, normal, or beta distribution. For this reason, we cannot compare the shape of the estimated effort uncertainty distribution with the shape of the actual error distribution. Nevertheless, it may be useful to inspect how the actual effort values are distributed relative to the effort PIs and analyse obvious biases in the estimated uncertainty distribution. An example of an obvious bias is the situation where the actual effort is never close to the estimated minimum effort of the PI, but frequently close to, or exceeding, the maximum effort. For the purpose of this analysis we introduce the measure Actual effort Relative to PI (ARPI):

$$ARPI_i = (Act_i - PI\_midpoint_i) / (Max_i - Min_i), \text{ where } PI\_midpoint_i = (Max_i+Min_i)/2$$

ARPI is a measure of the distance between the actual effort and the midpoint of the effort PI, normalised by the width of the PI. ARPI uses the PI_midpoint as reference point instead of the estimate (Est). The use of the PI_midpoint, we believe, simplifies the interpretation of the ARPI values. The ARPI measure provides values close to -0.5 and 0.5 when the actual effort is close to the estimated minimum and maximum, respectively, and equals 0 when the actual effort corresponds to the PI_midpoint. An ARPI value outside the interval [-0.5, 0.5] means that the actual effort is outside the effort PI. Notice that ARPI values can be larger in situations where the actual effort is underestimated compared to situations where the effort is overestimated. This is due to a floor effect, as it is impossible to use less than 0 work-hours, but no corresponding upper limit for possible amount of work. For example, if an activity is estimated to require between 10 and 20 work-hours, the PI_midpoint will be 15 work-hours. While the actual value cannot be lower than 0 work-hours (ARPI = -1.5), it can exceed 30 work-hours and result in ARPI values beyond 1.5. This means that the median ARPI may be a better indicator of biased effort PIs than the mean ARPI.

Example: Assume an organisation where the HitRate of the effort PIs correspond well to the confidence levels and the median ARPI is 0.3. This ARPI-value suggests that the effort PIs are biased towards optimism, i.e., that most actual effort values had been much closer to the maximum-value than the minimum-value of the effort PI. A consequence of this may be that the minimum-values should be increased in the subsequent estimates.

We have not found any other studies that provide this set of measures to evaluate human judgement based PIs. Most human judgement based PI evaluations, e.g., the studies described in (Kahnemann, Slovic et al. 1982), have focused only on the hit rates. To our knowledge, the ARPI measure has not been applied in any other study of human judgement based PIs. However, there are similarities between the analyses based on the ARPI measure and, for example, the "Inter-quartile Fraction" (proportion of actual effort falling into the estimator's inter-quartile range) applied in, for example, (Connolly and Dean 1997). The need for a set of effort PI measures, not only the hit rate, reflects the situation that "*... no single measure can capture all aspects of the accuracy or quality of an assessed probability distribution*" (Henrion, Fischer et al. 1993).

3 Studies

Table 1 displays an overview of the studies reported in this paper. The descriptions of the studies in Section 3.1-3.4 follow the template: Motivation, Design, and Results.

Study	**Topic**	**Objects**	**Participants**	**Described in**
A	Observation of the effort PIs of software projects.	3 industrial software projects + 13 student projects.	Project members of the industrial and student projects.	Section 3.1
B	Experiment on how the types of experience and estimation process affect effort PIs.	One industrial project.	20 software professionals forming five estimation teams.	Setion 3.2
C	Experiment on the relation between confidence level (50%, 75%, 90%, and 99%) and effort PIs.	One programming task.	105 computer science students.	Section 3.3
D	Experiment on how software developers evaluate the usefulness of software project effort PIs when the actual effort is known.	Two estimation scenarios, where the estimation performances of hypothetical developers are compared.	37 software professionals.	Section 3.4

Table 1: Overview of the studies reported in this paper.

The rationale for the sequence of studies is that we first wanted to examine whether over-confidence was a problem in real software projects (Study A), then try to understand the *reasons* for the observed over-confidence (Studies B, C, and D). Study B examines the possibility that the over-confidence is related to *type of experience or estimation process*, Study C examines the possibility that the concept of *confidence level*, e.g., 90 % certain, is difficult to interpret for software estimators, and Study D examines the possibility that there are unfortunate *feedback mechanisms* that reward over-confidence.

3.1 Effort PIs of Software Projects (Study A)

Motivation: Before studying the phenomenon of too narrow (over-confident) effort PIs in more artificial (experimental) estimation situations we should be reasonably sure that this phenomenon can be observed in real software development projects.

Design: Typically, software developers and project managers provide effort PIs on project activities. Then, the total effort PI is based on a mainly mechanical combination of the minimum and maximum activity effort estimates. Consequently, to evaluate human judgment based effort PIs we need to investigate the activity effort PIs. Surprisingly, it turned out to be difficult to find industrial projects where the actual use of effort could be compared to their activity effort PIs. We investigated more than 100 projects from 6 development organisations and found that about 30% of the projects had applied effort prediction intervals on the activity level. However, only 3 of these projects had logged the actual effort applying the same work break-down structure as used when estimating the effort. In fact, even these three projects had to split and combine some of the logged effort data to enable an analysis of the accuracy of the effort PIs. This paucity of feedback data is in itself an important finding. Without feedback on the effort PIs it is difficult to learn from experience, and easy to remain over-confident in one's own predictions. There was no formal confidence level imposed on the three industrial project's effort PIs. The effort PIs were simply formulated as minimum and maximum effort values and that it should be very unlikely that the actual effort values were outside the PIs. The estimation process was typically that the project leader asked the developers to provide the effort estimates and PIs of their own tasks. Then, the project leader adjusted the estimates and effort PIs, if he or she believed that they were unrealistic.

In addition to the 3 industrial projects, we examined the effort PIs of 13 software development projects conducted by computer science students during the autumn of 2001. The student projects were part of a course in software engineering at the University of Oslo. Each project involved 3-5 participants and lasted about 3 months. All participants estimated the most likely effort, and the 90% confidence PI of each activity, as part of the project planning work. During the project execution they logged the effort spent on each activity. When the projects were completed the estimated and actual effort of each activity were reported to the course management as part of a final project report. The estimation process of the student project was typically based on team-work, i.e., the participants of a project agreed on the effort PIs of the different project activities. This indicates that the estimation process of the students may have been more team based than that of the software professionals.

The total effort spent on the 3 industrial projects was approx. 400, 500 and 1800 work-hours. The deviation between the estimated and actual total effort was 15%, 6% and 13%. The effort PIs were available for 12, 18 and 19 project activities, i.e., in total 49 project activities were analysed. The effort spent on the 13 student projects varied from 119 to 453 work-hours with a mean project effort of 260 work-hours. The total effort was reasonably accurately estimated, with a mean estimation error of 27%. The total effort estimates of the student project were unbiased, i.e., there was no tendency of too low or too high effort estimates. The mean number of activities per student projects was 15, i.e., in total 15 * 13 = 195 project activities were analysed.

The inclusion of both industrial and student data enabled us to analyse differences in effort PIs between more and less experienced software developers.

Results:

HitRate: The industrial projects had hit rates of the activity effort PIs of 33%, 44% and 21%. Clearly, a meaningful interpretation of minimum and maximum effort should exclude hit rates as low as observed, i.e., there was a strong overconfidence observed in the effort PIs of the industrial projects. The hit rates of the student projects varied from 100% to 40% with an average hit rate of 62%. Only 3 out of the 13 student projects had a hit rate of more than 75%.

Consequently, all the industrial and most of the student projects had far too low hit rates. This indicates that the previous results on overconfidence on small student programming tasks, see (Connolly and Dean 1997), are also valid for many larger software development projects. There is, of course, a need for an analysis of more industrial projects to draw a general conclusion about the existence and magnitude of overconfidence in industrial projects. Optimally, one should have a random selection from the population of industrial software development projects. However, the poor performance of the 3 reported industrial projects, together with the lack of prediction interval feedback found in our examination of industrial projects suggest that overconfidence is present in most industrial software development projects.

PIWidth: The PIWidth of the activities of the industrial project varied from 0.2 to 2.1, with a mean value of 0.5. The student project activity PIWidths varied from 0.2 to 3.0, with a mean value of 1.1. The median PIWidth of activities with actual effort inside the effort PI was similar to the median of those with actual effort outside the effort PI for both the industrial project (0.51 versus 0.54) and the student projects (1.05 versus 1.14), i.e., activities with wider effort PIs did not have a higher hit rate. An explanation consistent with this finding is that the variation of PIWidth reflects the underlying uncertainty, whereas the level of overconfidence is unchanged. For example, if a project consistently provides 50% too narrow effort PIs, then the PIWidth values of the activities including and excluding the actual effort probably will be similar. If this explanation is correct, it would be possible to improve the judgment based effort PIs through simple extensions, e.g., multiplying the minimum and maximum estimates with a person and/or project specific constant value to achieve the desired confidence level. However, this procedure would also lead to very wide effort PIs, probably not accepted by the project. For example, one of the industrial projects would need an average activity effort PIWidth of approx. 3.0, as opposed to the actual average of 0.35, to include about 90% of the actual efforts.

PIWidth-BRE correlation: In the industrial projects there were no significant correlations between interval width and estimation accuracy. The correlation values per project were 0.3, 0.1 and -0.2. Similarly, the student projects showed no significant correlations between interval width and estimation accuracy. The individual correlation values of the student projects varied from -0.2 to 0.5.

Different activities have different levels of uncertainty, leading to a variation in PIWidth. This human judgment based variation in PIWidth seems, however, not to be a meaningful indicator of the accuracy of the estimated most likely effort.

ARPI: Figures 1 and 2 display histograms of ARPI values of the project activities of the 3 industrial projects and the 13 student projects. Only the activities with ARPI in the interval [-0.5, 0.5] have actual effort inside the effort PI (see explanation in Section 2.) While the industrial projects had a weak bias towards pessimism (actual activity effort lower than the minimum value), the student projects had a weak bias towards optimism (the actual activity effort higher than the maximum value). These patterns are, however, weak and cannot exclude unbiased activity effort PIs.

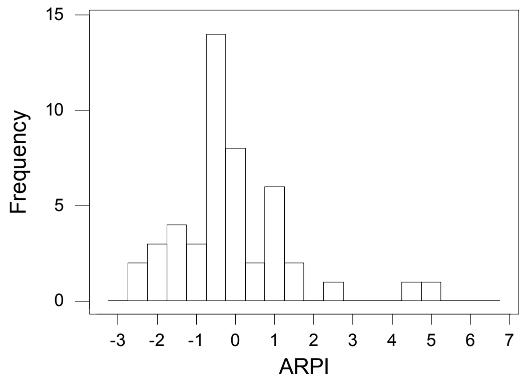

Figure 1: ARPI scores of activities of 3 industrial project (2 outliers excluded)

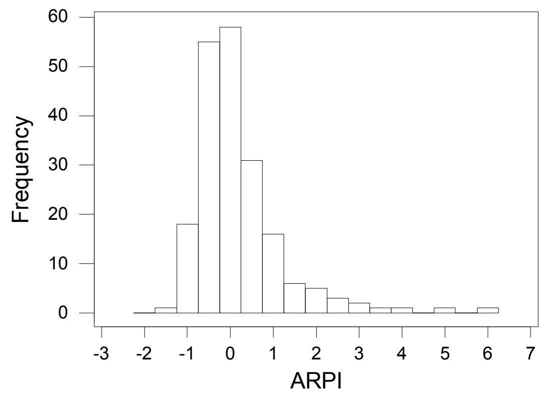

Figure 2: ARPI scores of activities of 13 Student projects

In sum, the results of Study A confirm that both professionals and students tend to produce too narrow effort PIs of software development activities, leading to hit rates far below that expected from the stated confidence levels. In this study, the student project effort PIs had higher hit rates than the industrial projects. Thus, apparently, the students did a better uncertainty estimation job than the professionals. A possible reason for this is that the students knew very well that that their knowledge about the required work was very limited and that nobody "expected" narrow effort PIs. The software professionals may, on the other hand, feel a pressure to indicate high development skills through narrow effort PIs. We examine and discuss this explanation in Study D. Another possible reason for the better performance of the students is that they were taught about effort PIs and overconfidence. We interviewed a number of the professional project managers when surveying projects applying effort PIs. The interviews revealed that there was a low awareness of too narrow effort PIs.

3.2 Impact from Role and Team Work on the Total Effort PI (Study B)

Motivation: The accuracy of effort PIs may depend on type of project role and experience as well as characteristics of the estimation process, e.g., individual versus team processes. Information about effects of these variables may give valuable input on how to understand and improve the effort PIs.

Design: Twenty software professionals from the Norwegian division of a large international e-commerce software development organisation were allocated to five estimation teams, with four participants in each. Each participant spent about 2 work-hours on the estimation tasks. The company was paid for the participation in the study. This allowed us to select highly skilled and motivated software professionals with experience from all relevant estimation roles. Thus, each estimation team consisted of one "Engagement manager" (responsible for contract and customer relations), one Project manager, one Software developer, and one User interaction designer. Of these, the project managers and software developers were technical experts, who could be assumed to know much about the programming tasks, whereas the engagement managers were supposed to have more experience with contract negotiations and user needs, and the user interaction designers knew most about human-computer interaction and graphical design of the system. The median length of total software professional experience was 4.8 years; median experience in current role was 3 years.

As input to the estimation process the participants received the requirement specifications of a project previously completed by the organisation, but not known to the participants. The original estimate of this project was 1240 work-hours. However, this was not known by the participants. This project was a typical project for the organisation and the specification used in the experiment was the one they had received from the customer. The estimation process of the experiment, including the four estimation roles, was similar to the estimation processes typically used by the organisation, and the participants were told to behave as if this was a real estimation task.

The actual effort of the specified project had been about 2400 work-hours, i.e., about twice as high as the original project estimate. This actual effort value cannot, however, be taken on face value as a "normative" right answer of the task given to the estimation teams. A software project has several possible outcomes and repeated completions of the same project, even if we assume no learning, would lead to a distribution of different effort values. For example, the original project could have met unexpected problems that the project estimated by the participants in our study might avoid. Discrepancies between "actual" and "estimated" values should accordingly be interpreted with some caution. The question of whether the actual effort of the project is within the 90% PI is, we believe, more meaningful. A 90% PI should include most unexpected problems encountered by a project. An interview with the project leader of the actual project indicated that there had been no exceptional project problems.

The estimation process was divided into two parts. In Part 1 the participants individually estimated the most likely total effort (ML) and a 90% PI of the project effort without discussion with the other members of the team. This was an unusual task for the software developers and the user interaction designers, who typically were only asked to estimate their own work. They knew, however, how large their part of a project typically would be, e.g., that programming and unit test would be about 50% of the total effort, and used this knowledge to derive a total effort estimate. Subsequently, in Part 2 of the experiment, the participants belonging to the same team discussed their estimates to agree on a common team estimate, based on the group's total experience and estimation skills. Part

2 was taped on a video-recorder.

Results: Table 2 displays the individual and group effort estimates (ML = Estimate of most likely effort, Min = Minimum estimate of effort, Max = Maximum estimate of effort, 90% effort PI = [Min, Max]).

ROLE:	**Engagement Manager Estimates**			**Project Manager Estimates**			**User Interaction Designer Estimates**			**Software Developer Estimates**			**Team Estimates**		
	ML	**Min**	**Max**	**ML**	**Min**	**Max**	**ML**	**Min**	**Max**	**ML**	**Min**	**Max**	**ML**	**Min**	**Max**
Team 1	1200	660	1740	960	770	1150	1500	1000	2000	960	900	1200	1100	900	1500
Team 2	1550	1000	4000	1820	1400	2200	1140	1010	1592	585	400	700	1500	1200	2500
Team 3	1850	1400	2300	300	268	332	1260	910	1630	220	110	400	1550	1300	1900
Team 4	547	470	630	914	850	1100	620	58	840	660	440	920	1339	1205	1473
Team 5	2286	1143	3429	984	802	1181	1500	1000	2000	900	720	1080	2251	1126	3377
Median value	1550	1143	2300	960	802	1150	1260	1000	1630	660	440	920	1500	1200	1900
Mean values	1487	935	2420	996	818	1193	1204	900	1612	665	54	860	1548	1146	2150

Table 2: Effort estimates from the members of five teams in Study B

General characteristics: The individual estimates of most likely effort varied from 220 (!) to 2286 work-hours with a mean value of 1088 work-hours. All these 20 values fall below the actual value of 2400 work-hours. The mean estimate is close to the original estimate of 1240 work-hours, indicating a realistic estimation process. The time required for this project was, once again, strongly underestimated. But not all participants were biased to the same extent. The lowest effort estimates were provided by the developers and the project managers, whereas the user interaction designers and the engagement managers gave generally higher estimates. The team estimates were in most teams higher than the average individual estimates, the exception being Team 1, where the team estimates are close to the mean of the individual estimates. The mean team estimate was 1548 work-hours.

HitRate: Based on an actual effort of 2400 work-hours, the overall hit rate of the individual effort PIs was as low as 10% (2 out of 20). The engagement managers had a hit rate of 40%, the user interaction designers 0%, the project managers 0% and the software developers 0%. The team PI estimates had a hit rate of 40%. Similar to Study A, we find a strong bias towards too narrow effort PIs.

PIWidth: The individual PIWidths varied from 0.2 to 1.9 with a mean value of 0.6. The engagement managers had a mean PIWidth of 0.7, the user interaction designers 0.5, the project managers 0.3 and the software developers 0.5. The mean PIWidth of the team estimates was 0.5, i.e., the increased hit rate of the team estimate was caused by an improved estimate of most likely effort and not wider effort PIs. The PIWidths were similar to those of the industrial project described in Study A.

PIWidth-BRE correlation: The correlation between the BRE (estimation accuracy) and PIWidths was 0.05 and not significant, i.e., results similar to those reported in Study A. Interestingly, the correlation between mean BRE and mean PIWidth of a role is strongly negative (-0.9), i.e., the roles with least accurate estimates had the narrowest intervals! There are, however, too few data to use this as more than a weak indication of a relationship.

ARPI: The ARPI values are depicted in Figure 3. None of the participants had minimum values higher than the actual effort (ARPI < -0.5). Most of the participants had maximum values lower than the actual effort (ARPI > 0.5). There was consequently a strong bias towards too optimistic maximum values.

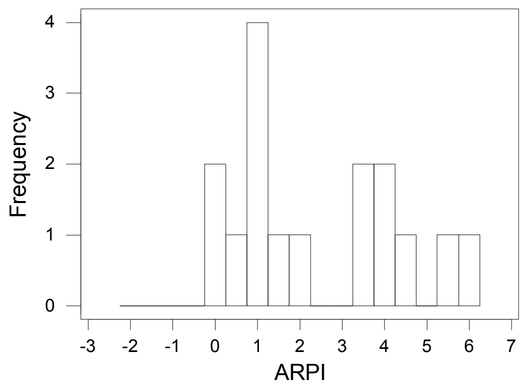

Figure 3: ARPI distribution – individual estimates (excludes four outliers)

One of the most interesting observations was that, in spite of (or because of?) their non-technical background, the engagement managers and user interaction designers provided the best effort PIs. A potential reason for this finding is the strategy followed when estimating the effort. The software developers, due to their technical knowledge, provided a much more detailed work break-down structure compared with the engagement managers and user interaction designers. It is difficult to know in advance all activities, features and implementation uncertainty, and detailed work break-down estimates may consequently fail to include activities that are more automatically included when estimating effort based on larger project units, e.g., when searching for other projects similar to the project to be estimated. As a result, technical background did not lead to better effort PIs, only to more confidence in the estimates. This is in line with the distinction between an "inside" versus an "outside" view in predictions (Kahneman and Lovallo 1993). Predictors working from an inside view try to take into account the separate components of the phenomenon in question, running the risk of missing out important parts or unforeseen problems. From an outside view, it is easier to take a statistical attitude asking, for instance, what have been normal completion efforts of such tasks in the past. A failure to include all activities may also explain the difference in ARPI results between Study A and B. In Study A we found that activity effort ARPIs were almost unbiased, while in Study B we found a strong bias of the total effort ARPI towards optimism. If the reason for the ARPI bias in Study B is the omitted activities, then the activity effort ARPIs of that study may still be unbiased, i.e., correspondence with the results in Study A.

Another interesting observation is that the most successful team, Team 5, seems to have used a different strategy than the other teams, ending up with minimum - maximum estimates that simply were the most likely effort plus minus approx. 50%. This seems due to two team members, the engagement manager and the user interaction designer who already have used this strategy for their individual estimates (in the other groups, this strategy seems to be used only by the user interaction designer of Team 1). This suggests that simple rules-of-thumbs may be useful to avoid too narrow effort PIs.

Although we tried to keep the estimation task as realistic as possible, the participants knew that this was a research study and not a real estimation task. When we discussed the issue of realism with the participants they emphasised that in a real estimation task they would contact the customer for more information when the specification was unclear. However, the participants were encouraged to incorporate all types of uncertainty in the effort PIs. Consequently, higher specification interpretation uncertainty should lead to wider effort PI, not necessarily to lower hit rates. The study observes the PI estimates of one single project, and more studies are needed to show the generality of the findings to other types of projects, e.g., more predictable projects.

3.3 Different Confidence Levels and Interval Width (Study C)

Motivation: Interviews with a number of the participants of Study A or B indicated that they did not use much thought on confidence level (probability) when they were asked to provide minimum and maximum effort values. For example, they did not seem to seriously consider the difference between 70%, 90%, or 99% confidence when estimating the minimum and maximum effort. Some of the software professionals described their minimum and maximum effort values as absolute boundaries, i.e., 100% confidence PIs, whereas others referred to their PIs as "almost certain" intervals. Thus the subjective interpretation of effort PIs may deviate strongly from the intended statistical interpretation. Study C tests one basic statistical assumption of effort PIs: Higher confidence levels should lead to wider effort PIs.

Design of study: The participants in Study C were students taking an introductory university course in programming. The students were asked to provide an effort estimate of own programming work based on a brief requirement specification similar to programming tasks they had solved earlier, i.e., the students had some experience upon which they could base their effort PIs. The students were given a brief introduction to confidence levels and effort PIs, and were then divided into four groups. The only difference between the groups was the stated confidence level of their effort PIs. Group 1 participants were asked to base their effort PIs on a 50% confidence level, whereas the confidence levels of Groups 2 to 4 were 75%, 90%, and 99%, respectively.

The variation in effort estimates, from less than 10 to more than 1000 work-hours, indicate that the students differed greatly in programming skills and/or how they interpreted the programming task. The students did not carry out the programming task, so their estimation and PI accuracy could not be assessed. This is, however, not a limitation of this study since we were only interested in the relation between the interval width (PIWidth) and the confidence level.

Results: The results are described in Table 3.

Group	**Confidence Level**	**Number of Students**	**Median PIWidth**
1	50%	28	0.8
2	75%	26	0.7
3	90%	24	0.7
4	99%	27	0.7

Table 3: Prediction intervals for estimates based on different confidence levels

With increasing confidence level, PIWidth should obviously increase. If we assume that the actual effort values of the estimated project is normally distributed and that the correct PIWidth for a 50% confidence level is 0.8 (as reported by participants in Group 1 see Table 3), the PIWidth should be about 1.4 for a 75% confidence level, 2.0 for a 90% confidence level, and 3.1 for a 99% confidence level. The actual variation of PIWidth was, however, close to zero!

Our results indicate that, in situations similar to the one in our experiment, estimators do not consider the task of providing an effort PI with 50%, 75%, 90% or 99% differently. Clearly, they do not attend to the probabilities when producing minimum and maximum values. The results should, however, be interpreted very carefully. For example, the results do not imply that software developers are unaware of the difference between being 99% and 50% certain. Probably, most software developers would provide wider effort PIs for 99% than for 50% confidence levels if the same individuals were asked to provide both effort PIs for the same development task.

As discussed in Study A there are important differences between students and software professionals. Consequently, we need more realistic estimation tasks completed by software professionals to establish the general validity of these findings. However, since most software professionals seem to lack feedback on their effort PIs (see Study A) and do not get any training in interpreting and providing effort PIs, we believe that software professionals will suffer from the same interpretation problems as students when assessing the confidence of their own estimates.

3.4 Effort PI Interpretations (Study D)

Motivation: The interviews with the software developers indicated an "unwillingness" to provide sufficiently wide effort PIs under conditions of acknowledged high uncertainty. One software developer explained: *"I feel that if I estimate very wide effort PIs, this will be interpreted as a total lack of competence and has no informative value to the project manager. I'd rather have fewer actual values inside the minimum-maximum interval, than providing meaningless, wide effort PIs."*

In other words, the observed problems of providing sufficiently wide effort PIs and the lack of sensitivity to the confidence level may be caused by the estimators' desire to show competence and to provide useful information. Study D was designed to investigate whether managers share these concerns. If both producers and consumers of effort PI estimates want them to be narrow and "informative" rather than wide and accurate, inaccuracies will not be sanctioned, and there will be little motivation to improve hit rates.

Design of study: A total of 37 software professionals in a Norwegian organisation developing financial and e-commerce applications participated in this study, which took part during a course in software effort estimation. All of them had some experience in project (or sub-project) management. Following a short training in effort PIs and confidence levels, the participants were presented with two scenarios and asked to answer questions connected to each scenario. Both scenarios were introduced as follows:

Assume that you are the manager of a software development project and have asked two developers (D1 and D2) to estimate most likely effort, minimum effort and maximum effort for the same 5 development tasks. D1 and D2 have been instructed to be 90% confident (certain) that the actual effort they will use is between the minimum and maximum effort. The table (Table 4 for Scenario 1 and Table 5 for Scenario 2) *displays the estimates of D1 and D2 and the actual effort used. Assume that this information is all you have about developer D1 and D2 and the tasks.*

Development Task	**Developer**	**Most Likely Effort**	**Estimated Minimum**	**Estimated Maximum**	**Confidence Level**	**Actual Effort**
1	D1	4	2	6	90%	6
2	D1	10	8	12	90%	5
3	D1	2	1	3	90%	2
4	D1	1	0,5	2	90%	4
5	D1	10	8	15	90%	15
1	D2	4	1	10	90%	6
2	D2	10	2	20	90%	5
3	D2	2	0,5	5	90%	2
4	D2	1	0,5	3	90%	4
5	D2	10	5	30	90%	15

Table 4: Scenario 1 (D1 and D2 are equally confident, but produce different intervals)

Development Task	**Developer**	**Most Likely Effort**	**Estimated Minimum**	**Estimated Maximum**	**Confidence Level**	**Actual Effort**
1	D1	4	3	6	90%	8
2	D1	10	7	12	90%	7
3	D1	2	1	3	90%	2
4	D1	1	0,5	2	90%	4
5	D1	10	8	14	90%	13
1	D2	4	3	6	60%	8
2	D2	10	7	12	60%	7
3	D2	2	1	3	60%	2
4	D2	1	0,5	2	60%	4
5	D2	10	8	14	60%	13

Table 5: Scenario 2 (D1 and D2 produce equal intervals, but report different confidence)

Following each scenario table, the participants were asked to answer the following questions:
a) As a project manager, would you prefer the estimates of D1 or D2?
b) Do you believe D1 or D2 knew more about how to solve the development tasks?
c) Do you believe D1 or D2 knew more about the uncertainty of the effort required to solve the task?
The possible answers were for each task "D1", "D2", "no difference" and "don't know".

Explanation of the scenarios: In Scenario 1, D1 and D2 have the same actual effort, same confidence level (90%) and make the same estimates of most likely effort. The only difference is that D2's effort PIs are much wider than the corresponding estimates of D1. As a result, D1 has a hit rate of only 3 out of 5 (60%), while D2's hit rate is of 4 out of 5 (80%), i.e., D2's hit rate is in better correspondence with the confidence level of 90% than is the hit rate of D1.

In Scenario 2 D1 and D2 have the same hit rate (60%), actual effort, estimated effort, and effort intervals. The only difference is that D1 has a confidence level of 90%, whereas D2's confidence level is only 60%. D2 has, consequently, a perfect correspondence between confidence level and hit rate, while D1 is strongly over-confident.

Both scenarios should lead to a preference of D2 in response to question (a) if a statistically correct interpretation of PIs is applied. D2 should also be chosen in response to question (c), as the developer who knows more accurately the degree of uncertainty involved. For question (b) – who has more task knowledge – there is no single normative answer.

Results: The frequency of preferences connected with Scenario 1 and 2 are displayed in Table 6 and 7.

Scenario 1 **D1 and D2 Equally Confident**	**D1 (narrow intervals)**	**D2 (wide intervals)**	**No Difference**	**Do Not Know**
(a) Preferred developer	35	2	0	0
(b) Knowledge of task	29	1	2	5
(c) Knowledge of uncertainty	11	17	2	7

Table 6: Answers Scenario 1

Scenario 2 **D1 and D2 Equal Intervals**	**D1 (high confidence)**	**D2 (low confidence)**	**No Difference**	**Do Not Know**
(a) Preferred developer	15	14	3	5
(b) Knowledge of task	15	11	4	7
(c) Knowledge of uncertainty	3	23	3	8

Table 7: Answers Scenario 2

In Scenario 1 the preference for D1 was very clear for the (a) and (b)-questions, while there was a weak preference for D2 in the (c)-answers. The preference for D1 in the (a)-question means that nearly all the software professionals preferred the much too narrow intervals of D1 to the statistically more appropriate intervals of D2! To eliminate the possibility that the participants had misunderstood the question, e.g., that they based the (a)-answers on the performance of the effort PIs and not the development abilities of D1 and D2, we conducted a few informal interviews with the software professionals. There were, as expected, a large variance in how minimum, maximum and confidence levels were interpreted, but the scenario and the questions were reasonable well understood.

Answers to question (b) revealed that wider intervals were believed to indicate less task knowledge. At the same time, most software professionals believed that D2 knew more about the uncertainty of the task (question (c). This implies that knowledge about how to solve a task and knowledge about the uncertainties involved are viewed as two completely different (even opposite) types of knowledge.

In Scenario 2 there was no clear preference for D1 or D2 with respect to question (a). From a normative point of view, one should have expected a clear preference for D2, who had a perfect hit rate and used the same intervals as D1. This means that strong over-confidence is not universally condemned. Interestingly, 11 of those 15 who preferred the estimates of D1 in the (a)-question answered that they believed that D2 knew most about the uncertainty of the required effort in the (c)-question. However, the estimate of the uncertainty of the required effort was the only described difference between D1 and D2. A potential explanation of this surprising combination of answers is that the software professionals evaluated the effort PIs of D1 to be "more correct" when correct, i.e., that D1's correct effort PIs were based on more knowledge and better analyses compared with D2.

In sum, the answers to question (a) indicate that software professionals do not generally prefer the most statistically correct effort PI estimator. At the same time, there is a difference between the two scenarios indicating that wide PIs are viewed as less acceptable than low confidence ratings. This indicates that if we changed the uncertainty elicitation process to a process where the estimators provided the confidence level of a fixed-width effort PI, for example an interval where the minimum is 50% and the maximum 200% of the most likely effort, the realism may increase. In (Jørgensen and Teigen 2002c) we describe an experiment where this change in elicitation process did remove the overconfidence bias. Evaluation of this strategy for reduction of effort PI biases in more realistic software development effort estimation situations is an interesting topic for further study.

The use of the width of the effort PIs as an indicator of knowledge about how to solve the task, question (b), is consistent with the finding in Study B, where the role with the narrowest effort PIs, the software developers and the project managers, also had most knowledge about how to develop the software.

The unwillingness and/or inability to provide sufficiently wide effort PIs described in the previous studies becomes more understandable in light of the preferences of other software professionals. Through narrow effort PIs, a software developer is more likely to be perceived as knowledgeable about how to solve the task, even when the intervals are much too narrow, or the confidence is much too high. In more realistic situations than our scenarios, the preference for narrow intervals may be even stronger than we observed, because there are frequently no feedback processes that enable the discovery of over-confidence. The preferences pattern found in this study is, we believe, an important obstacle for realistically wide effort PIs.

4 Discussion and Related Work

The studies described in this paper and other related studies show that judgment-based, high confidence, PIs are, on average, much too narrow., see for example the human-judgment studies (Alpert and Raiffa 1982; Lichtenstein, Fischhoff et al. 1982; McClelland and Bolger 1994; Wright and Ayton 1994; Bongaarts and Bulatao 2000). There is, in our opinion, no reason to believe that expert estimation of software development effort PIs should not be subject to the same phenomenon. Surprisingly, awareness of this problem seems to be very low in the organisations we have studied, as well as in textbooks on project management. For example, the PERT project management textbook (Moder, Phillips et al. 1995) assumes without further discussion that people are able to provide high confidence project activity minimum and maximum values. Similarly, we found only one research paper on human judgement based software development effort PIs (Connolly and Dean 1997). This lack of research focus and organisational awareness may have had several unfortunate consequences, for example:

- Other goals than effort PI accuracy dominate the uncertainty estimation process, i.e., the goal of appearing skilled and providing "information". In other words, there is a difference between the intended and the actual interpretation of the task of estimating effort PIs. The results we report in Study D support this.

- The development of supporting tools and procedures to improve effort PI estimation processes has been neglected. For example, the only formal models of software development effort PIs we have found are described in very recent papers (Angelis and Stamelos 2000; Stamelos and Agnelis 2001; Jørgensen and Sjøberg 2002b) and have not as yet been tested in realistic software project effort estimation.
- Project managers frequently get an overly optimistic view of the uncertainty of the estimates and project plan. This may lead to sub-optimal decisions. For example, if the maximum efforts are severely underestimated, as in Study B, necessary actions to reduce risk will not be taken.
- Too little attention is devoted to improve feedback mechanisms and to learn to assess uncertainty (Jørgensen and Sjøberg, 2002a).

The phenomenon of too narrow prediction intervals has frequently been discussed along with the more general phenomenon of overconfidence in predictions. There has been a number of studies on overconfidence, see Arkes (2001) for a recent overview of mainly forecasting studies on this topic. Interestingly, there are a few groups of people that are reported to avoid overconfidence, namely weather forecasters, expert bridge players, and horse race bettors. These groups have several features in common: They provide a large number of uncertainty predictions of similar phenomena, and they get immediate feedback on their predictions, which is recorded and systematically compared with their actual performance. In other words, relevant, systematic feedback on previous predictions is a requirement for provision of proper effort PIs. Software professionals do not, as far as we have experienced, get uncertainty prediction feedback at all, or if they do, only incompletely and at a late stage when the project is finished. At this stage important estimation assumptions may have been forgotten. If feedback is received weeks after the effort PIs were developed, discrepancies can easily be attributed to external factors like changes in project scope, rather than to poor original effort PIs. In this way the software professionals may persist in believing that their effort PIs are much better than they really are. In situations where no official effort PIs are given and the evaluations of the uncertainty assessments are conducted after the completion of the project, there is an even greater danger of hindsight bias. A number of human judgment studies, e.g., (Fischhof 1975), show how the lack of explicit predictions can obstruct the learning process. Typically, people strongly overestimated their actual ability to predict the outcome.

Besides the lack of feedback, we believe that the following reasons are important for the inaccurate effort PIs:

- *Interpretation difficulties.* When historical or statistical data are sparse, software professionals have to base their uncertainty assessment on task knowledge, i.e., their knowledge about how to solve the task. It is, however, difficult to see how an accurate confidence level can be derived from such knowledge. For example, how can a software developer know that it is 90% certain that the actual effort is between 100 and 200 work-hours without information about the distribution of similar tasks? The difficulty people have when dealing with uncertainty level is demonstrated in Study C, where variations in confidence level had no impact on the widths of the PI.
- *Inside versus outside perspective:* Individuals with some task knowledge appear to prefer a breakdown of the task into sub-tasks (the internal perspective) to the process of using information from completed similar tasks (the external perspective). When the uncertainty is high, however, the internal perspective may easily lead to omitting sub-tasks and, consequently, too optimistic effort PIs. The results of Study B are in line with this explanation. In Study B, the effort PIs of those with less task knowledge were

more accurate than those with more knowledge on how to solve the task.

- *Hidden agendas.* Software professionals have goals other than just a high correspondence between confidence level and hit rate. In particular, the desire to be evaluated as a skilled software developer may be an important agenda that leads to overly narrow effort PIs. As shown in Study D, a narrow effort PI is, indeed, applied as an important predictor of skills.
- *Project managers favour narrow intervals and high confidence.* Study D indicates that most software developers preferred effort PIs that were much too narrow or based on a much too high confidence level, even when they *knew* that they were much too narrow or too high. Similar results for probability prediction preferences were found by (Keren and Teigen 2001). Keren and Teigen interpret this preference for overconfident to more accurate predictions as manifestations of the "search for definitive predictions" principle. Transferred to our effort PI context, this principle implies that project managers prefer high confidence predictions, i.e., narrow intervals or high certainty, because high confidence predictions make their project planning and execution easier. Project managers have to make decisions and they prefer decisions based on high confidence, i.e., definitive, predictions of the required effort.

Combining these reasons, we see that narrow effort PIs are understandable, and in some ways even rational. Without historical data and feedback, a meaningful interpretation of the confidence level of effort PIs is difficult. Without a meaningful interpretation of effort PIs, wide PIs become less attractive than narrow PIs, which are associated with some obvious benefits like positive evaluation of development skills and easier project decisions.

Through our measures of interval width (PIWidth), correlation between interval width and estimation accuracy (BRE) and the distribution of ARPI values we found several other interesting patterns, such as:

- Increased knowledge about how to solve the task led to narrower intervals, but not necessarily higher estimation accuracy or effort PIs with better correspondence to the stated confidence level (Study B).
- There was no clear correlation between relative interval width and estimation accuracy (Study A and B). This result is different from the results described by Yaniv and Foster (1997), where participants were asked to estimate the most likely, minimum and maximum value of various quantities (so-called almanac question quantities). Yaniv and Foster found a correlation between interval widths and estimation accuracy, i.e., that wider intervals correlated with less accurate estimates. However, they used the *absolute* accuracy and interval widths rather than the *relative* accuracy and interval widths, as we have used in the present studies. It should come as no surprise that the absolute discrepancies between actual and estimated values and the PIWidths are larger for large quantities than for smaller quantities. We believe that our analyses of the relative width give a more interesting indication of the use of interval width as a predictor of estimation accuracy.
- The ARPI-distributions suggest that too high minimum values may be just as frequent as too low maximum effort values for project *activities* (Study A and B). However, too low maximum effort values may be more frequent for the total effort PIs. This finding is consistent with the situation were the main source of over-optimism is omitted activities, not under-estimated activities.

The student-part of the results in Study A, together with related studies, e.g., the software development effort study described in (Connolly and Dean 1997) and the human judgment (almanac questions) study described in (Alpert and Raiffa 1982), indicate that an improvement of the effort PIs requires much more than a simple lecture based training. Instead it may be necessary to change the estimation process, provide tool support and include training opportunities. One type of training is so-called “deliberate practice” (Ericsson, Krampe et al. 1993), i.e., activities especially designed to improve specific aspects of an individual’s performance. In our context this may mean that software professionals should from time to time practice estimation tasks as “training tasks”. Then, the effort estimates and PIs, the estimation assumptions, the project events relevant for the actual use of effort, etc. should be carefully documented. When the project is completed, a thorough evaluation of the estimates should be conducted. However, even such practice may not in itself be sufficient for efficient learning when based on improper learning models. In addition, software professionals should be able to realise the “probabilistic” nature of software development efforts, see (Brehmer 1980) and (Hammond 1996) for general discussions on this human judgment topic. Proper interpretations of probability and confidence levels may, thus, be necessary in order to learn from experience in a highly probabilistic environment.

An alternative to human judgement based effort PIs are PI models which are based on simple rules-of-thumb. An example of this is provided by NASA’s Software Engineering Laboratory (SEL) (NASA 1990), see Table 8.

Time of Estimation	**Suggested Effort Prediction Intervals**
End of the requirements definition and specification phase	[Estimate / 2.0, Estimate · 2.0]
End of the requirement analysis phase	[Estimate / 1.75, Estimate · 1.75]
End of the preliminary design phase	[Estimate / 1.40, Estimate · 1.40]
End of the detailed design phase	[Estimate / 1.25, Estimate · 1.25]
End of the implementation phase	[Estimate / 1.10, Estimate · 1.10]
End of the system testing phase	[Estimate / 1.05, Estimate · 1.05]

Table 8: NASA SEL’s guidelines for estimation of effort prediction interval

This type of effort PI model is simple and can be calibrated to any type of organisation and projects. The use of it would clearly have increased the hit rate compared with the human judgement based effort PIs in Study A and B. If the rule-of-thumb is not based on an organisation’s own experience, it may be difficult to trust, i.e., an organisation should develop its own effort PI rules-of-thumb. A simple tool for this purpose is described in (Jørgensen and Sjøberg 2002b).

Another alternative to pure human judgement based effort PIs is the combination of experts and models. Experts and models have different strengths and weaknesses, which could ideally be complementary. Models are, for example, unaffected by organisational pressures, and not biased towards over-optimism. Experts can, on the other hand, identify new variables relevant for the actual project effort PIs, and can identify “broken-leg” situations, i.e., very rare situations that are not normally included in an effort PI estimation model. Benefits from combining models and expert judgment in business forecasting are, for example, reported in (Blattberg and Hoch 1990).

5 Conclusions and Further Work

The fact that people tend to provide too narrow prediction intervals has been known in the human judgment research community since the 1970s. The first report on this topic was written already in 1969 by Alpert and Raiffa, see (Alpert and Raiffa 1982). Surprisingly, this over-confidence knowledge may not yet have reached most text-books and courses on project management.

The results described in this paper may contribute to the existing knowledge on too narrow PIs in several ways. The results confirm that the phenomenon of too narrow PIs also applies to effort estimation of software projects (Study A and B), and provide elements of an explanation of the reasons for too narrow software development effort PIs. Possible reasons for over-confidence in effort PIs are, amongst others: Lack of immediate and meaningful feedback (see Study A), the difference between the "inside" versus "outside" view (see Study B), problems with the interpretation of confidence level (see Study C), hidden agendas related to appearing skilled (see Study D), and that project managers reward over-confidence by software development estimators (Study D). The examination of the relation between interval width and estimation suggests that the interval width should not be used as an indicator of estimation accuracy (see Studies A, B and C).

We may be able to remove the strong bias towards over-confidence through better feedback, "deliberate training", uncertainty elicitation processes based more on an "outside" view, or, combination of effort prediction intervals from different sources. We have on-going or planned studies focussing on all these improvement approaches.

References

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

Angelis, L., Stamelos, I., 2000. A simulation tool for efficient analogy based cost estimation. Empirical software engineering 5, 35-68.

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

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

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

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

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

Conte, S. D., Dunsmore, H. E., Shen, V. Y., 1986. Software engineering metrics and models, Benjamin/Cummings Publishing Company Inc.

Dawes, R. M., 1988. Rational choice in an uncertain world, Harcourt Brace & Company.

Ericsson, K. A., Krampe, R. T., Tesch-Römer, C., 1993. The role of deliberate practice in the acquisition of expert performance. Psychol. Rev. 100 (3), 363-406.

Fewster, R., Mendes, E., 2001. Measurement, prediction and risk analysis for Web applications. International Software Metrics Symposium, London, England, IEEE Computer Society, Los Alamitos, Calif., 338-348.

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.

Gray, A., MacDonnell, S, and Shepperd, M., 1999. Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment. *Sixth International Software Metrics Symposium*, IEEE Comput. Soc, Los Alamitos, CA, USA, 216-227.

Hammond, K. R., 1996. Human judgement and social policy. New York, Oxford University Press.

Henrion, M., Fischer, G. W., Mullin, T., 1993. Divide and conquer? Effects of decomposing on the accuracy and calibration of subjective probability distributions. Organizational Behavior and Human Decision Processes 55, 207-227.

Höst, M., Wohlin, C., 1998. An experimental study of individual subjective effort estimations and combinations of the estimates. International Conference on Software Engineering, Kyoto, Japan, 322-339.

Jørgensen, M., 1997. An empirical evaluation of the MkII FPA estimation model. Norwegian Informatics Conference, Voss, Norway, Tapir, Oslo, 7-18.

Jørgensen, M., Sjøberg, D. I. K., 2001. Impact of software effort estimation on software work. Journal of Information and Software Technology 43, 939-948.

Jørgensen, M., Sjøberg, D. I. K., 2002a. Learning from experience in a software maintenance environment. Journal of Software Maintenance 14, 123-146..

Jørgensen, M., Sjøberg, D. I. K., 2002b. An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy. Journal of Information and Software Technology (in press).

Jørgensen, M., Teigen, K. H., 2002c. Uncertainty Intervals versus Interval Uncertainty: An Alternative Method for Eliciting Effort Prediction Intervals in Software Development Projects. Proceedings of the International Conference on Project Management (ProMac), Singapore, 343-352.

Jørgensen, M., Moløkken, K., Situational and Task Characteristics Systematically Associated With Accuracy of Software Development Effort Estimates, Accepted for publication at the IRMA 2003 conference.

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

Kahnemann, D., Slovic, P., Tversky, A., 1982. Judgement under uncertainty: Heuristics and biases, Cambridge University Press.

Keren, G., Teigen, K. H., 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.

Känsälä, K., 1997. Integrating risk assessment with cost estimation. IEEE Software (May/June), 61-67.

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

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

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

Miyazaki, Y., Takanou, A., Nozaki, H., Nakagawa, N., Okada, K., 1991. Method to estimate parameter values in software prediction models. Information and Software Technology 33 (3), 239-243.

Miyazaki, Y., Terakado, M., and Ozaki, K., 1994. Robust regression for developing software estimation models. Journal of Systems Software 27 (1), 3-16.

Moder, J. J., Phillips, C. R., Davis, E. W., 1995. Project management with CPM, PERT and precedence diagramming. Wisconsin, U.S.A, Blitz Publishing Company.

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

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

Stamelos, I., Agnelis, L., 2001. Managing uncertainty in portfolio cost estimation. Information and Software Technology 43, 759-768.

Wright, G., Ayton, P., 1994. Subjective probability. West Sussex, England, John Wiley.

Yaniv, I., Foster, D. P., 1997. Precision and accuracy of judgmental estimation. Journal of behavioral decision making 10, 21-32.