Uncertainty Intervals versus Interval Uncertainty: An Alternative Method for Eliciting Effort Prediction Intervals in Software Development Projects

Jørgensen, M.[1]
Simula Research Laboratory
P.O. Box 134, NO-1325 Lysaker, Norway
magne.jorgensen@simula.no
and
Teigen, K. H.
Department of Psychology,
University of Oslo,
P.O. Box 1094 Blindern, NO-0317 Oslo, Norway
k.h.teigen@psykologi.uio.no

Abstract: Frequently, there is a poor correspondence between the judged and the actual uncertainty of effort usage in software development projects. This may to some extent be a consequence of the uncertainty elicitation process. Traditionally, software developers are asked to provide the minimum and maximum effort of development work for a given confidence level, e.g., minimum and maximum effort that includes the actual effort usage with a 90% probability. An alternative uncertainty elicitation process is to instruct the software developers to provide the uncertainty of a given effort interval, e.g., the probability that the actual effort is between 50% and 200% of the estimated most likely effort. In an empirical investigation, this alternative process led to significant improvement of prediction interval accuracy. The observed improvement using this alternative elicitation process can, we believe, be explained through a simplified interpretation of historical prediction accuracy data, less "conflicting estimation goal", and less influence from the "anchoring effect".

Keywords: Effort estimation, prediction intervals, empirical study

1. MOTIVATION

Software development projects have a bad reputation for effort overruns. According to a large survey carried out by Standish Group in the 1990s, see http://www.standishgroup.com/visitor/chaos.htm, the average software project effort overrun was as high as 189% of the original estimate, and only 17% of the projects completed on-time, on-budget and with all features and functions as initially specified. The characteristics of software development projects, such as "one of a kind", complex development environments and changing requirements, mean that we cannot always expect accurate effort estimates even with the best estimation models and processes. In other words, both from an empirical and analytical point of view we should frequently expect a large uncertainty regarding the actual use of effort in software development projects.

Knowledge about the size of this uncertainty is generally of interest. For example, the project budget allocated to resolution of unexpected problems should depend on the level of uncertainty of the effort estimates (McConnel 1998). In addition, knowledge about the level of uncertainty of an effort estimate is necessary for proper learning from and analysis of the deviation between the estimated and the actual effort (Jørgensen and Sjøberg 2002b). A large deviation between the estimated and the actual effort does not necessarily indicate poor estimation skills; a large deviation may be due to high uncertainty.

1.1 Prediction Intervals and Over-confidence

The uncertainty of effort estimates can be described through effort prediction intervals. For example, a project leader may estimate that the most likely effort of a development project is 10000 work-hours and

[1] Corresponding author

that it is 90% certain 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. This use of effort prediction intervals is similar to the use of activity duration prediction intervals in the statistical PERT (Program Evaluation and Review Technique) approach (Moder, Phillips et al. 1995).

The statistical definition of a prediction interval, assuming a single random sample of n independent observations from a normally distributed population, is (Christensen 1998):

$$\bar{y} \pm t(conf, df) \cdot s \cdot \sqrt{1 + \frac{1}{n}} \qquad \text{E1}$$

Where is the sample mean, *t(conf,df)* is the t-value for confidence level *conf* and *df* degrees of freedom, and *s* is the standard deviation of the sample. Here, the *1/n*-term reflects the uncertainty of the mean value and the *1*-term the standard deviation of the population. Consequently, the width of an effort prediction interval depends on how well we know the mean use of effort on similar projects and the (inherent) variance in use of effort of such projects. In real software projects there are also other factors that impact the width of the effort prediction interval, see Jørgensen and Sjøberg (2001) for an overview. These factors are, however, unimportant for the analyses in this paper.

The statistical assumptions and definition of prediction intervals are, probably, too complex to be used as a basis for human judgment based effort prediction intervals. This complexity may be one reason for the poor correspondence between the confidence level (judged uncertainty) and proportion of actual effort values inside the prediction interval (actual uncertainty). A confidence level of *K*% should, on the long run, result in a proportion of actual values inside the prediction interval (hit rate) of *K*%. However, Conolly and Dean (1997) reported that the actual effort used by software developers to solve programming tasks fell in only 60% of the tasks inside their 98% confidence effort prediction intervals. 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% confidence. Other studies on human judgment report that this type of over-confidence is found in most domains. Arkes (2001) gives an overview of recent studies on over-confidence.

The problems of knowing how to calculate effort prediction intervals may be an important reason for inaccuracy, but do not explain why there is an over-confidence, and not an under-confidence or an unbiased variation. Potential reasons for this over-confidence are:

- *Estimation goals in "conflict" with the estimation accuracy goal.* In particular, the software professionals' goals of appearing skilled and provide "informative" prediction intervals, may be in conflict with the goal of sufficiently wide prediction intervals (Yaniv and Foster 1997; Keren and Teigen 2001).
- "*Anchoring effect".* The "anchoring effect" is described as *"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). Several studies, e.g., (Kahnemann, Slovic et al. 1982; Jørgensen and Sjøberg 2002a), report that people typically provide estimates much too close to the anchor value and that they are not sufficiently aware of this influence from the anchor. The estimate of the most likely effort may easily become the anchor of the estimate of minimum and maximum effort. Consequently, the minimum and maximum effort will not be sufficiently extreme.

Formal software development effort prediction interval models do not have the interpretation and over-confidence problems described above. Surprisingly, the research and use of formal software development prediction interval models have been very limited. The only software development effort prediction interval models we have found are very recent (Angelis and Stamelos 2000; Jørgensen and Sjøberg 2002c), and we have not found any reported use of such models in realistic estimation contexts. The lack of evaluated and commercially available software development effort prediction interval models may, however, not be the only reason for the lack of use. A preliminary comparison of judgement and model based effort prediction intervals, carried out by one of the authors of this paper, indicates that software professionals make better use of the available uncertainty information than the formal models. The formal prediction intervals models seem to have a very good correspondence between confidence level and hit rate, but sometimes they provide apparently meaninglessly wide effort prediction intervals. Given the same hit rate, the formal models provided, on average, much wider prediction intervals than the software professionals. Important uncertainty information, e.g., information about the developer who is supposed to complete the task, is difficult to integrate in formal models. In other words, the need for "useful" effort prediction intervals, not only correspondence between confidence and hit rate, may favour human judgement based prediction intervals.

1.2 Learning from Experience

It is difficult to learn from previous interval predictions when applying the traditional uncertainty assessment elicitation process. Assume, for example, that a software developer estimates that the required effort of completing a programming task is 50 work-hours and is asked to provide a 90% confidence prediction interval. The developer is 90% confident that the actual effort is between 40 and 60 work-hours. When the task is completed, the developer, however, found that he had used 80 work-hours. What is the proper learning from this outcome? Probably, the estimated maximum effort was too low. On the other hand, the confidence level was 90%, not 100%, i.e., one out of ten actual effort values was expected to be outside the prediction interval. Perhaps this was the "exception" and that there is no need for wider effort prediction intervals. To assess the need for wider prediction intervals, the developer has to compare the prediction intervals with the actual effort of *several* similar tasks. However, learning is still difficult. Assume that the developer compares the prediction interval with the actual effort of several similar tasks and finds that the 90% confidence prediction intervals included the actual effort in only 60% of the tasks. This indicates a need for wider prediction intervals, but *not how much*. For this purpose, the software developer needs more sophisticated analyses, e.g., an analysis of the percentage change of minimum and maximum effort values needed to include 90% of the actual effort values. This type of analysis is not necessarily very difficult, but may require calculation tools and careful analyses.

1.3 Prediction Interval Elicitation Processes

We have interviewed more than 20 project managers that were familiar with effort prediction intervals, but found none that used an explicitly defined model to establish minimum and maximum effort values. The calculation of and learning from minimum and maximum effort estimates seemed to be based on rather intuitive models. According to our interviews, the elicitation of software development effort prediction intervals are based on the following general process (denoted TRADITIONAL in this paper):

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

This paper evaluates the effect of swapping step 2 and 3, i.e., to change the elicitation process to the following process (denoted ALTERNATIVE in this paper):

1. Estimates the most likely effort.
2. The minimum and maximum effort are calculated as fixed proportions of the most likely effort. In our study we based the proportions on the software development effort prediction interval guidelines from NASA (1990), that is, we set the minimum effort to 50% and the maximum effort to 200% of the most likely effort.
3. Determine the confidence level, i.e., the probability that the actual effort is between the minimum and maximum effort.

Formally, the difference between these two processes is not large and the software development effort prediction interval models described in (Angelis and Stamelos 2000; Jørgensen and Sjøberg 2002c) can easily be changed to cover the ALTERNATIVE process. However, from a learning perspective there may be important differences. The described problems with the TRADITIONAL elicitation process may, to some extent, be removed. Assume that a project leader is asked to provide the confidence level of a 50%-200% effort prediction interval of a software project and that there exists a number of similar, previously completed, projects that have been estimated applying this 50%-200% fixed-width effort prediction interval. A simple analysis of the previous prediction intervals gives, for example, that 70% of those intervals included the actual effort. Consequently, given that the history is relevant, the expected probability that the actual effort is included by the new 50%-200% prediction interval is about 70%. As opposed to the TRADITIONAL process, no calculation tools are needed.

Since the minimum and maximum effort values are calculated "mechanically", it may also be easier to be realistic about the probability that the actual effort is inside the interval, i.e., the over-confidence may be reduced. While minimum and maximum values provided by oneself, as in the TRADITIONAL process, may be used to signalize skill, those provided automatically may reduce the feeling of interval ownership and consequently increase the focus on realism. In addition, as opposed to the TRADITIONAL process, there is no obvious anchor value that influences the prediction intervals toward over-confidence.

1.4 Hypothesis and Measures

Based on the differences between the two elicitation processes, we propose the following hypothesis:

Software developers learn more efficiently from previous prediction interval experience and have a better correspondence between confidence level and hit rate when applying the ALTERNATIVE process than when applying the TRADITIONAL process.

Here, the hit rate is measured as (E2):

$$HitRate = \frac{1}{n}\sum_i h_i, \quad h_i = \begin{cases} 1, & \min_i \le Act_i \le \max_i \\ 0, & Act_i > \max_i \vee Act_i < \min_i \end{cases}, \qquad \text{E2}$$

where min_i and max_i are, respectively, the minimum and maximum values of the prediction interval of task *i*, Act_i is the actual effort of task *i* and *n* is the number of estimated tasks. Note that a correspondence of mean confidence level and hit rate only *indicates,* not guarantees, a high accuracy. For example, assume that an estimator is supposed to provide 90% confidence prediction intervals. Instead, the estimator in 9 out of 10 estimates provides extremely wide prediction intervals to be 100% sure to include the actual effort. In 1 out of 10 estimates he provides a very narrow prediction interval that is likely to exclude the actual effort. This strategy is likely to result in a 90% hit rate, i.e., a very good correspondence with the required confidence level. There are similar problems with confidence levels of the ALTERNATIVE process. A perfect correspondence may, consequently, hide very large inaccuracies on single task level. Murphy and Winkler (1970) call this correspondence between hit rate and confidence level 'secondary validity', while the 'primary validity' is defined as the correspondence between the prediction interval and the underlying probability distribution for an *individual* estimation task. In software effort estimation we do not have access to the 'primary validity' of a single prediction interval and must therefore rely on 'secondary validity' evaluations. Hence, it is important to assess whether some estimators follow strategies that optimize the 'secondary validity' on the cost of the 'primary validity'.

To test our hypothesis we carried out an empirical study with professional software developers. The design of this study is described in Section 2 and the results in Section 3. Section 4 briefly discusses the validity of the findings and Section 5 concludes.

2. DESIGN OF STUDY

2.1 Subjects

Twenty-nine software developers and project managers of a Norwegian, e-commerce software development organization were paid about 100$ per hour to participate in the experiment. The participants were, randomly, divided into two groups: TRADITIONAL and ALTERNATIVE. The participants of the TRADITIONAL group used the TRADITIONAL elicitation process and those of the ALTERNATIVE group used the ALTERNATIVE elicitation process, see description in Section 1.3.

2.2 Tasks

The estimation work was completed in a Microsoft Excel spreadsheet, i.e., the necessary calculations were easy to carry out. The experiment was divided into two parts:

1) TRAINING: Instructions and training in use of the estimation spreadsheet and how to provide effort prediction intervals. As training tasks the effort and the effort prediction intervals of 10 real-world software development projects were estimated based on an "experience database" of 5 similar projects. Feedback was given after each estimate. When the estimates were completed, the software developers were asked to analyse and reflect on their own performance, in particular the correspondence between confidence level and hit rate. The average time spent on the training was approx. 30 minutes.
2) ESTIMATION: Estimation of 30 software enhancements tasks previously conducted in a large telecom company. Task 1 was estimated based on an "experience database" including 5 previously completed tasks; Task 2 was estimated based on the 5 tasks and feedback, i.e., the actual effort, on the Task 1 estimate; Task 3 was estimated based on the 5 tasks and the feedback on the Tasks 1 and 2 estimates, and so on. The average time spent on the estimation tasks was approx. 90 minutes.

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

Table 1 shows a partly completed spreadsheet (translated from Norwegian) of one participant from the ALTERNATIVE group. The spreadsheet of the TRADITIONAL group included the same tasks, but instead of a fixed interval (50%-200% of the most likely effort) width, there was a fixed confidence level (90%). The first five tasks constitute the initial experience database and were not estimated by the participants.

Table 1: Example spreadsheet from the ALTERNATIVE process

ESTIMATION OF SOFTWARE ENHANCEMENT TASKS

Project Id	Estimated lines of code	Type of development task	Most likely effort (work-days)	Minimum effort	Maximum effort	Confidence level	Estimate completed (y or n)	Control of input	Actual effort (work-days)	Estimation deviation	Actual effort included in the prediction interval?
1	550	New module	*x*	*x*	*x*	*x*	*x*	*x*	**5**	x	x
2	50	Interface change	*x*	*x*	*x*	*x*	*x*	*x*	**4**	x	x
3	20	Interface change	*x*	*x*	*x*	*x*	*x*	*x*	**1**	x	x
4	10	Control flow change	*x*	*x*	*x*	*x*	*x*	*x*	**0,7**	x	x
5	250	Interface change	*x*	*x*	*x*	*x*	*x*	*x*	**5**	x	x
6	500	New module	5,0	2,5	10,0	70 %	y	OK	25	-0,8	No
7	500	New module	10,0	5,0	20,0	50 %	y	OK	10	0,0	Yes
8	400	Interface change	15,0	7,5	30,0	50 %	y	OK	20	-0,3	Yes
9	100	Interface change	10,0	5,0	20,0	60 %	y	OK	5	1,0	Yes
10	250	Control flow change	10,0	5,0	20,0	70 %	y	OK	20	-0,5	Yes
11	75	New module	4,0	2,0	8,0	50 %	y	OK	2	1,0	Yes
12	1000	New module	40,0	20,0	80,0	60 %	y	OK	7	4,7	No
13	200	Control flow change	10,0	5,0	20,0	70 %	y	OK	4	1,5	No
14	300	New module	5,0	2,5	10,0	50 %	y	OK	2	1,5	No
15	500	New module	10,0	5,0	20,0	50 %	y	OK	5	1,0	Yes
16	10	Control flow change	2,0	1,0	4,0	50 %	y	OK	0,5	3,0	No
17	40	New module	4,0	2,0	8,0	60 %	y	OK	1	3,0	No
18	100	Assignment change	5,0	2,5	10,0	20 %	y	OK	1	4,0	No
19	300	New module	4,0	2,0	8,0	40 %	y	OK	5	-0,2	Yes
20	15	Control flow change	1,0	0,5	2,0	60 %	y	OK	1	0,0	Yes
21	400	Control flow change					n				
22	400	Control flow change					n				
23	30	Assignment change					n				
24	170	Control flow change					n				
25	100	Assignment change					n				
26	50	Interface change					n				
27	150	Control flow change					n				
28	84	Interface change					n				
29	30	Control flow change					n				
30	20	Control flow change					n				
31	350	New module					n				
32	200	Control flow change					n				
33	400	Interface change					n				
34	900	Assignment change					n				
35	75	Interface change					n				

As can be seen in Table 1, the information about the development tasks was very limited. Only the estimated size (lines of code) of the task to be programmed and the type of task were included. We reduced the available information to these two variables because they were the only variables that were significantly correlated to the development productivity (measured in lines of code per work-hour).

There are some factors that are different from a realistic estimation task. For example, in most realistic cases the estimators would have more relevant information about the application, e.g., the complexity of the particular part to be changed, about the particular developer who carried out the task and about the necessary programming steps. This means that we cannot expect very accurate predictions of the actual effort. Our goal is, however, to analyse differences between the ALTERNATIVE and the TRADITIONAL process regarding their uncertainty assessment, not to study the accuracy of the prediction of most likely effort.

3. RESULTS

Three of the participants who followed the TRADITIONAL elicitation process did not understand the task of estimating the prediction intervals properly and were removed from the data set, leaving 12 software professionals in the TRADITIONAL group and 14 in the ALTERNATIVE group. To study the progress of the correspondence between hit rates and confidence, we divided the 30 tasks into three sets: Task 1-10, Task 11-20 and Task 21-30. There was no significant difference in the accuracy of the estimate of the most likely use of effort, i.e., there was no difference in estimation skills of the two groups. Both groups had a median deviation between the estimated and actual effort of about 50%. Table 2 indicates important differences in the correspondence between mean hit rates (Hit rate) and confidence levels (Conf.) for the two elicitation processes.

Table 2: Mean hit rate vs. mean confidence level (all participants)

	Training tasks		Task 1-10		Task 11-20		Task 21-30	
Group	Hit rate	Conf.	Hit rate	Conf.	Hit rate	Conf.	Hit rate	Conf.
TRADITIONAL	0.58	0.90	0.64	0.90	0.70	0.90	0.81	0.90
ALTERNATIVE	0.83	0.83	0.67	0.73	0.69	0.71	0.73	0.71

The participants who followed the TRADITIONAL process approached very slowly an acceptable correspondence between mean hit rate and confidence level. The participants who followed the ALTERNATIVE process, on the other hand, had a close to perfect correspondence between mean hit rate and confidence level on all groups of tasks.

The values shown in Table 2 do not indicate the variance in performance or whether the differences in performance are significant. For this purpose we apply the one-sided t-test on the difference in mean deviation between hit rate and confidence level for the two elicitation processes. This test show that there is only a significant difference between the two groups for the Training tasks (p=0,006) and Task 1-10 (p=0,02). For Task 11-20 (p=0,57) and 21-30 (p=0,26) the difference is not significant. The individual hit rates and confidence levels are displayed in Table 3. Pairs of hit rate and mean confidence level with deviation larger than or equal to 0.2 are printed in bold.

Table 3: Hit rate vs. mean confidence level (individual data)

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

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

expected improvement through combination of prediction intervals from the TRADITIONAL approach is much lower.

4. DISCUSSION

In real software development work, there are rarely as much as 20-30 similar tasks that can be used as basis for new estimates. This means that the performance of the 10 first estimation tasks, based on a few earlier tasks, may be the most important criterion for comparing the two estimation process. The experiment indicates that the ALTERNATIVE elicitation process gives significantly more accurate prediction intervals on the first tasks than does the TRADITIONAL process. Our hypothesis, stated in Section 1.4, is therefore supported. It is nevertheless, an interesting result that also the TRADITIONAL processes led to reasonable accurate prediction intervals given a long sequence of feedback on previous prediction intervals.

The finding that the participants who followed the ALTERNATIVE process were not over-confident is to some extent surprising. Most human judgment studies, e.g., Hora et al. (1992), report that over-confidence is robust to variation in elicitation methods. There have, however, not been many studies comparing the TRADITIONAL with the ALTERNATIVE elicitation process. In fact, the only study on this topic we have been able to find is from 1978 (Seaver, von Winterfeld et al. 1978). That study reports results similar to ours, i.e., a strong reduction in over-confidence using the ALTERNATIVE process. The results described by Seaver et al. (1978) have been subject to methodological objections by other researchers. Conolly and Dean (1997) suggest that the findings were not valid and *"presumably reflecting the cueing effect of the experimenter's range* (minimum – maximum) *setting"*. This objection cannot be made to the minimum and maximum values of the ALTERNATIVE process in our study since the participants knew that these values were calculated automatically as 50% and 200% of the estimate of the most likely effort. Neither do we agree that the results described by Seaver et al. (1978) reflect a cueing effect. For example, some of the minimum-maximum values were randomly selected numbers between 1 and 99%.

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

- Only one level of confidence (TRADITIONAL process) and one interval width (ALTERNATIVE process) were studied. These levels were chosen because they were typical (90% confidence) or recommended (50%-200% prediction intervals). It is, nevertheless, possible that other confidence levels and interval widths would lead to other results. For example, it is possible that the selection of a 70% confidence level would remove the bias towards over-confidence applying the TRADITIONAL process, and that an 80%-120% prediction interval would lead to over-confidence applying the ALTERNATIVE process. In other words, the improvements we found may depend on the chosen levels. For example, Seaver et al. (1978) found no large difference between the two processes for the "inter-quartile" interval, i.e., the interval that should reflect a 50% confidence level.
- The estimation process of the participants in the experiment was different from their normal process. Normally, they would know more about the task, the development environment, etc. There is, therefore, a need for a replication of the study in more realistic estimation situations. We have now started a study comparing the TRADITIONAL and ALTERNATIVE process in real industrial software development estimation processes.
- We only use 'secondary validity' measures comparing prediction interval accuracy, see Section 1.4. However, we found no evidence of strategies that were used to optimize the 'secondary validity' on the cost of the 'primary validity', i.e., this is probably not a major limitation.

5. CONCLUSIONS

The study described in this paper found that predicting the uncertainty of a fixed-width minimum-maximum effort interval (denoted the ALTERNATIVE uncertainty elicitation process) led to significantly better accuracy than the strategy of providing a minimum-maximum effort interval for a given level of uncertainty (denoted the TRADITIONAL uncertainty elicitation process). In addition, the bias towards over-confidence seems to be removed applying the ALTERNATIVE process, which means that a combination of uncertainty assessments from different predictors is likely to further improve the accuracy. The TRADITIONAL effort prediction interval elicitation process gave strongly over-confident prediction intervals. Both processes enabled learning from previous estimates. There was, however, a large difference in how quickly the learning occurred. While the ALTERNATIVE process led to an almost immediate correspondence between hit rate and confidence, the TRADITIONAL process approached an acceptable correspondence slowly. We believe that these differences can be explained through a much easier interpretation and learning process, less stimulus for over-confidence and avoidance of anchoring influence when applying the ALTERNATIVE process. The practical consequence of our findings is that software

development project managers should change from the TRADITIONAL to the ALTERNATIVE effort prediction interval elicitation process.

References

Angelis, L. and I. Stamelos 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. J. S. Armstrong, Kluwer Academic Publishers**:** 495-515.

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

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

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

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

Hora, S. C., J. A. Hora and N. G. Dodd 1992. Assessment of probability distributions for continuous random variables: a comparison of the bisection and fixed value methods. *Organizational behavior and human decision processes* 51: 133-155.

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

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

Jørgensen, M. and D. I. K. Sjøberg 2002b. Learning from experience in a software maintenance environment. *Journal of Software Maintenance (accepted for publication)*.

Jørgensen, M. and D. I. K. Sjøberg 2002c. A simple effort prediction interval method. *Achieving Quality in Information Systems (accepted for publication)*, Venize.

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

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.

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

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

Murphy, A. H. and R. L. Winkler 1970. Scoring rules in probability assessment and evaluation. *Acta Psychologica* 34: 273-286.

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

Seaver, D. A., D. von Winterfeld and W. Edwards 1978. Eliciting subjective probability distributions on continuous variables. *Organizational Behavior and Human Performance* 21: 352-379.

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