Research:

Impact of Experience on Maintenance Skills

Magne Jørgensen[*, 1,†] and Dag I. K. Sjøberg[‡,1]

[1]*Simula Research Laboratory, Oslo, Norway*

SUMMARY

This study reports results from an empirical study of 54 software maintainers in the software maintenance department of a Norwegian company. The study addresses the relationship between amount of experience and maintenance skills. The findings were, amongst others: (1) While there may have been a reduction in the frequency of major unexpected problems from tasks solved by very inexperienced to medium experienced maintainers, additional years of general software maintenance experience did not lead to further reduction. More application specific experience, however, further reduced the frequency of major unexpected problems. (2) The most experienced maintainers did not predict maintenance problems better than maintainers with little or medium experience. (3) A simple one-variable model outperformed the maintainers' predictions of maintenance problems, i.e., the average prediction performance of the maintainers seems poor. An important reason for the weak correlation between length of experience and ability to predict maintenance problems may be the lack of meaningful feedback on the predictions.

J.Softw. Maint: Res. Pract. Xxxx: yy:000-000

No. of Figures: 4. No. of Tables: 6. No. of References: z.

KEY WORDS: software maintainers, predictions, learning from experience, human judgement.

[*]Correspondence to: Magne Jørgensen, Simula Research Laboratory, P.O. Box 1080, N-0316 Oslo, NORWAY, Tel.: +47 22 84 00 59.
[†]Email: Magne.Jorgensen@simula.no
[‡]Email: dagsj@simula.no

1. INTRODUCTION

This paper reports a study on how the amount and type of software maintenance experience impact the frequency of unexpected major problems of a maintainer's work, and the accuracy of predictions of maintenance problems. The goal of our study is to increase the understanding of how much and when maintainers learn from experience. The results are based on a study of the software maintenance department of a large Norwegian company.

Experience is believed to have a large impact on the quality and productivity of software development and maintenance work. An illustration of this belief is the emphasis on the impact of experience on productivity in most software estimation models, e.g., (Albrecht 1979; Boehm 1981; Symons 1991), and the work on software experience databases, e.g., (Basili, Caldierea et al. 1994; Engelkamp, Hartkopf et al. 2000). Jones (1998) claims that application and staff experience are the two factors with strongest impact on maintenance productivity.

Despite this strong belief in the importance of experience, there has been little empirical research on how the amount and type of software maintenance experience impact the maintenance and prediction skills of the maintainers, i.e., the topics addressed in this study. The perhaps most relevant software studies on how experience impact maintenance skills were carried out in the 1980s. These studies indicate that the expert software developers are superior to novices in many ways. They implement changes more efficiently (Soloway, Adelson et al. 1988);[1] they organise the information in larger 'chunks' (McKeithen, Reitman et al. 1981); and they solve problems using more principle-based strategies (Weiser and Shertz 1983). These findings are in accordance with the findings in other studies comparing the cognition of experts with the cognition of less experienced professionals, e.g., (Ericsson and Lehmann 1996). It is, however, not obvious that the *amount* of experience is the main difference between experts and novices. Doane et al. (1990) report a rather weak

[1] These experienced programmers did, however, only outperform the novices when the programs they had to change followed the "rules of programming discourse". If, for example, the program had functions and variable names that did not correspond with the content or function, there was no significant difference between the experienced programmers and the novices.

correlation between amount of experience and skill. Instead, the amount of "deliberate practice", i.e., activities especially designed to improve specific aspects of an individual's performance may be more closely related to skill than amount of experience (Ericsson, Krampe et al. 1993). In other words, the *type* of experience or training may be more important than the amount of general experience. In addition to these studies there are a number of studies on the cognitive processes in software maintenance, e.g., (Hale and Haworth 1991; von Mayrhauser and Vans 1994; von Mayrhauser, Vans et al. 1998). The impact of experience on the cognitive processes is, however, not much emphasised in those studies.

Even if professionals' problem solving strategies and other cognitive processes may improve with increased amount of experience, the quality of their *predictions* may not improve (Camerer and Johnson 1991). In fact, most studies on the correlation between prediction quality and length of experience report disappointing results: *"Yet in nearly every study of experts carried out within the judgement and decision-making approach, experience has been shown to be unrelated to the empirical accuracy of expert judgements"* (Hammond 1996, p. 278). We have been unable to find any empirical software maintenance study on the relation between amount of experience and prediction ability.

The remainder of this paper is organised as follows. Section 2 gives a description of the maintenance department studied. Section 3 describes the measures, the data collection procedure and the hypotheses. Section 4 describes the data analysis, reports the results and discusses threats to the validity of the results. Section 5 describes related work. Section 6 discusses possible explanations for the results. Section 7 concludes and describes further work. Appendix 1 contains the data set used in the analyses.

2. THE ORGANISATION STUDIED

We studied 54, randomly selected, software maintainers in the maintenance department of a Norwegian company.[2] Below is a summary of the most relevant characteristics of the maintainers, the applications and the tasks.

The department maintained more than 70 software applications and employed about 110 developers and maintainers. The maintained applications were, typically, written in Cobol, 4GL or C and varied in size from a few thousands to about 500 000 lines of code (LOC). The age of the applications varied from less than a year to more than twenty years. The functions of the applications include payroll, order entry, billing and invoicing, inventory control, service management, network management and personnel administration. Most applications were used only by the company itself.

A maintenance task was allocated to only one person. The average length of maintenance and/or development experience of the maintainers was 7.7 years. Of this, on average, 3.4 years had been spent on maintenance on the application subject to the maintenance task. As much as 36 of the maintainers had participated in the development of the application maintained, i.e., the development of the first release or development of new releases of the application, and 21 had developed the particular part of the application changed in the maintenance task. In other words, the average experience level of the maintainers was rather high. The distributions of type of tasks, the technology used and the size of the applications were similar to what was found in studies such as (Lientz and Swanson 1980), and (Nosek and Palvia 1990), i.e., the organisation subject to our study is probably a typical maintenance organisation.

The maintenance process in the studied organisation followed, typically, the following steps:

Specify: Maintenance requests, written or oral, were given to the maintenance managers or the maintainers. Errors were corrected immediately, while other change requests were assigned a priority by a committee.

Plan: The maintenance managers and/or the maintainers planned the maintenance work based on earlier experience. No estimation or risk analysis model, tool support or historical data was available.

Design: The maintainers decided on a design and, for large tasks, wrote design documents.

Code: The coding was carried out in accordance with a common programming standard.

Test: The software changes were normally tested by the maintainer who had changed the software. The work of inexperienced maintainers was inspected and tested by more experienced maintainers.

Document: Most software changes were only documented in the code. More extensive documentation was postponed to more "quiet periods" or never carried out at all.

As far as we were informed, no inspection of the plan or the solution was required, but there were no formal obstacles against asking other maintainers to inspect their code or to investigate reasons for deviation from a plan. Typically, inexperienced maintainers were given extra attention from more experienced maintainers (mentors) the first 1-2 years, i.e., there was an informal training process for the most inexperienced maintainers. The risk analysis process was informal and based on expert judgement.

[2] This data set is a subset of a data set on maintenance tasks described in (Jørgensen 1995a). While the organisation's maintenance *tasks* were the population in that study, this study has the *maintainers* as its population. We, therefore, included only the *first* task solved by a maintainer from the original data set. This reduced the size of the data set from 109 maintenance tasks to 54 (one task per maintainer).

The organisation, work processes and applications we have studied are rather "traditional". Hence, there is a need for a replication of the results in more "modern" development and maintenance environments. However, as far as we have experienced, the challenges of learning maintenance skill and prediction ability from experience are quite similar in "traditional" organisations and in, for example, organisations developing and maintaining e-commerce applications. Although the technical environment may be very different, the properties of the maintenance process that enable or prevent learning from experience seem similar.

3. METHOD

3.1. Measures

defines the measures used in our study. Explanations, discussion and examples of each of the measures are given in Sections 3.1.1 – 3.1.5.

Table 1 about here, please.

3.1.1. Experience

There are, at least, two valid interpretations of experience, Oxford dictionary (1990): (a) *"event or activity that affects one in some way*", (b) "*knowledge or skill acquired from seeing and doing things*". The first interpretation describes experience as something that a person has seen or done, while the second interpretation emphasises the *learning* from what was seen or done. Our interviews indicate that the maintainers sometimes used the first interpretation, sometimes the second and sometimes a combination of them. For example, the experience level of a maintainer could in one context be described as closely connected to the number of years with relevant experience. In another context, the maintainers could state that two maintainers experiencing similar events and activities could reach the same experience level at very different points of time. Hence, both the number of years as professional and the skill level acquired could be important when determining experience level. The lack of a precise and common interpretation of maintenance experience points at a situation where it is difficult to ask the maintainers about the impact or importance of experience without identifying how experience is or should be interpreted. Clearly, if an interpretation of experience includes elements of maintenance skill, experience will be an indicator of less maintenance problems. This lack of precise definitions of experience may also be present in the software maintenance literature. For example, in spite of the importance put on application and maintenance experience in (Jones 1998), we find no guidelines on how to measure or interpret it.

For the purpose of this study we found the first interpretation of experience most useful and decided to measure experience in terms of MEXPTOT, MEXPAPP, DEVSYS and DEVMSYS, see . The following example illustrates the different types of experience measured: A maintainer with in total 6 years development and/or maintenance experience (MEXPTOT = 6 years) completes a maintenance task on an application. The maintainer participated in the development of that application (DEVSYS = Y), but not on the part of the application changed as part of the maintenance task (DEVMSYS = N). The developer has been responsible for maintenance of the application the last 2 years (MEXPAPP = 2 years). Note that the same maintainer may have different experience values on MEXPAPP, DEVSYS and DEVMSYS on another maintenance task. This reflects the realistic situation that a maintainer should be regarded as experienced on one task, and less experienced on another.

3.1.2. Confidence

When the maintainers had received and understood the task specification, but before they had started to design or program the solution, we interviewed each of them and asked: "Do you think you know how to solve the task?" The possible answers were: "Yes", "Maybe" and "No". These were our initial categories of task solving confidence. We urged them to answer "Yes" only if there was a very low risk of major unexpected problems. The interviews indicate only a small difference between the confidence levels "Maybe" and "No". For this reason, we use only two confidence classes in our confidence measure; Y ("Yes") and N ("Maybe" + "No"), see definition of CONFIDENCE in .

CONFIDENCE is in this study interpreted as the maintainer's prediction of major unexpected problems. For example, if the maintainer was sure about how to solve the task (CONFIDENCE = Y), this is considered equal to a prediction of no major unexpected problems.

3.1.3. Unexpected problems

The predictability of maintenance work is indicated by whether there have been major unexpected problems, see definition of UNEXPEC in . UNEXPEC was collected immediately after the task was completed asking the maintainer whether there had been any major unexpected problems when solving the task. It was important to ask immediately after the task was solved in order to avoid a "selective memory" of the maintainers. Our goal was to assure that the maintainers had the maintenance work fresh in mind, that they interpreted "major unexpected problems" similarly and that they did not try to hide problems.

In 1/3 of the tasks the maintainers had faced major unexpected problems, in 2/3 not. Most of the unexpected problems were of the following types:

- More complicated task than expected
- Unexpected ripple effects
- Changed/incomplete task requirements or communication problems with customer

3.1.4. Prediction accuracy

We wanted to compare different types of experience with prediction accuracy. Consequently, we needed a meaningful measure of prediction accuracy. Our measure (PRED) is defined in . The categories of PRED are "Too optimistic" (OPT), "Too pessimistic" (PES) and "Correct" (OK). We assigned the PRED-value OPT to a prediction if the maintainer had a high confidence (CONFIDENCE = Y) that there would be no problems and still major unexpected problems occurred. Similarly, we assigned the PRED-value PES if the maintainer expected maintenance problems (CONFIDENCE = N) and none occurred. Otherwise, the PRED-value was assigned OK (= correspondence between prediction and outcome).

There were in total 10 too optimistic (OPT) predictions, 13 too pessimistic (PES) predictions and 31 correct (OK) predictions. PRED may not be a very good measure of prediction accuracy. There may, for example, be a problem that different maintainers interpret "know how to solve the task" and "major unexpected problem" differently. On the other hand, the alternatives may not be better. A comparison of estimated effort with actual effort may have other disadvantages. The actual effort may, for example, be strongly influenced by the estimated effort, see (Abdel-Hamid and Madnick 1986) and (Jørgensen and Sjøberg 2001).

3.1.5. Size of task

Our measure of the size of a maintenance task (SIZE) is defined in . The average SIZE of a maintenance task was 206 LOC. On average, 141 LOC were inserted, 57 LOC updated and 8 LOC deleted.

3.2. Data collection

A random set of maintenance tasks was selected using the following algorithm:

Preparation: A list of names of maintainers in the organisation.
Collect data:
REPEAT
  SELECT a name (MaintainerName) at random from the list;
  Collect data about the next task to be *started* by MaintainerName[5]
UNTIL "sufficient number of tasks"

After the maintainer had understood the task specification, but before the design and programming were started, the task solving confidence (CONFIDENCE) was measured. Immediately after the task was finished the first author collected, through interviews and observations, the following data: MEXPTOT, MEXPAPP, DEVSYS, DEVMSYS, UNEXPEC and SIZE (see). The use of interviews was important to ensure consistent interpretation of important terms. As reported in (Jørgensen 1995b), questionnaires may not ensure consistency in interpretation of important maintenance terms, such as the types of activities that should be regarded as maintenance, in spite of a precise definition in the questionnaire.

The data used in the analyses described in this paper is enclosed in Appendix A.

3.3. Testing of hypotheses

3.3.1. The hypotheses

Based on our own experience as software developers and previous studies on experience we decided to hypothesise that more experience leads to less unexpected problems (H1) and that more experience does not lead to better predictions of maintenance problems (H2).

[5] Selecting from the on-going tasks, i.e., not the next task to be started, would result in an over-representation of large maintenance tasks.

H1 is tested through the following sub-hypotheses:
H1.1: The median length of experience (MEXPTOT) is lower on tasks with major unexpected problems (UNEXPEC).
Explanation: Intuitively, more experience as a maintainer should lead to less unexpected problems. If this is true, tasks with major unexpected problems are more frequently carried out by inexperienced maintainers, i.e., the median length of experience is lower on tasks with major unexpected problems.
H1.2: The median length of application experience (MEXPAPP) is lower on tasks with major unexpected problems (UNEXPEC).
Explanation: As reported in Section 1, application experience is believed to be the most important predictor of maintenance skills. If application experience is important, then there should be a higher proportion on maintainers with short application experience on tasks with major unexpected problems and, consequently, a lower median length of application experience.
H1.3: The proportion of major unexpected problems (UNEXPEC) depends on whether the maintainer participated in the development of the application (DEVSYS and DEVMSYS) or not.
Explanation: A deep understanding of the design rationales and design logic of an application may require participation in the development of the application. H1.3 tests whether this type of experience is important for the degree of maintenance problems.

H2 is tested through the following sub-hypotheses:
H2.1: The median length of experience (MEXPTOT) is independent of the prediction accuracy category (PRED).
Explanation: Earlier studies, see Section 1, report no or low correlation between length of experience and prediction ability. In particular, the ability to learn from previous uncertainty (risk) predictions may be poor, see for example (Conolly and Dean 1997). Based on these studies we hypothesise that a longer total experience do *not* lead to higher proportions of correct predictions.
H2.2: The median length of application experience (MEXPAPP) is independent of the prediction accuracy category (PRED).
Explanation: H2.1 tests whether more application dependent experience improves the prediction ability.
H2.3: The prediction accuracy (PRED) is independent of whether the maintainer has participated in the development of the application (DEVSYS and DEVMSYS) or not.
Explanation: H2.3 tests whether the knowledge acquired through participation in the initial development improves the prediction ability.
The null-hypotheses are the negations of the hypotheses H1.1 – H2.3.

A more exploratory part of our study is the analysis of whether a "simple" prediction model would predict major unexpected problems just as well as the maintainers. If a simple prediction model is better or has a performance similar to the expert based predictions, this would suggest that simple rules should replace or support the predictions of software maintainers and that there are serious flaws in the learning and/or prediction process of the maintainers. The analysis on this topic is described in Section 4.3.

3.3.2. Statistical tests

Tests: The hypotheses are tested using the non-parametric rank based statistics Mann-Whitney U and Kruskal-Wallis, and the nominal scale based Chi-square test (Wonnacott and Wonnacott 1990). We use these non-parametric tests because we cannot assume that the measured experience values follow a normal distribution: experience has a lower limit of 0 years of experience, and, typically, a long tail towards larger values.

P-value: The number of statistical tests has an impact on how the p-values should be interpreted. There are several techniques for dealing with this, see (Rosenberg 1996) for a practical introduction. Holm (Holm 1979), for example, describes a process of adjusting the required significance level to reflect the number of tests in a *family of tests*. A family of tests should, according to (Miller Jr 1981), be interpreted as the tests carried out as an individual experiment. Holm's process sets the adjusted significance level equal to $alpha/(K-i+1)$, where $i = 1,...,K$ is the index of each test ordered by the p-value, $p_1 <= p_2 <= ... <= p_K$, and *alpha* is the 'initial' significance level. The initial significance level should reflect the significance level required if only one hypothesis were tested. This adjustment means that p_1, the smallest p-value, must be compared with $alpha_1 = alpha/K$. The largest p-value p_K must be compared with $alpha_K = alpha$. Testing is stopped when a non-significant result is obtained. All untested hypotheses are then considered non-significant.

We conduct 8 tests in our study[6] and should, to be conservative, require that the most significant result is 1/8 of the p-value level required if only testing one hypothesis, the second most significant result is 1/7 of the p-

[6] H1.3 and H2.3 contain two tests each.

level value required, etc. For example, if we initially set *alpha* to 0.1, then the lowest p-value should be less than 0.1/8 = 0.013 to be significant. In our analyses it is difficult to provide arguments for a specific pre-determined significance level (*alpha*). Therefore, as suggested in (Wonnacott and Wonnacott 1990), we present the actual p-values for each test and use this p-value, together with other relevant information, in an argumentation for or against rejection of the hypotheses. In particular, the argumentation includes the significance level adjustments suggested by Holm.

Size of effect: The p-value of hypothesis testing only indicates the presence of a difference, not the size of the difference. To measure the size of the difference between two groups, we use Cohen's size of effect measures *d* and *w* (Cohen 1977). For ratio scaled variables, *d* is used; for nominal scale values, *w* is used. Conventionally, a *d* of about 0.2 indicates a small size of effect, a *d* of about 0.5 indicates a medium size of effect and a *d* of about 0.8 or more indicate a large size of effect. Similarly, a *w* of 0.1 / 0.3 / 0.5 indicates a small/ medium / large size of effect (Cohen 1977). We present the size of effect only when there is a significant difference.

Power: The power of a test indicates the probability of detecting a significant relationship if there is one. This is highly relevant information for non-significant results. As pointed out in (Cohen 1962) and (Sedlmeier and Gigerenzer 1989), many empirical studies have a too low power for relevant effects of size to draw meaningful conclusions on the basis of non-significant p-values. For most non-significant tests we will, thus, indicate the power for some relevant effects of size. The power tests are based on a significance level of 0.1. We use the statistical tool NCSS, module PASS, to calculate the power values.

4. RESULTS

This section tests all hypotheses described in Section 3.3.1 and discusses threats to the validity of the results. A discussion of the results in the context of related work (Section 5) is given in Section 6.

4.1. Unexpected problems (H1)

H1.1: The median length of experience (MEXPTOT) is lower on tasks with major unexpected problems (UNEXPEC)

A display of properties of the distribution of MEXPTOT for each UNEXPEC-category is shown in a so-called violin plot[7] in . The violin plot is made by combining a box plot with two vertical density distributions. One density trace extends to the left; the other to the right. There is no difference in these density traces other than the direction in which they are extended. The violin plot highlights the peaks and valleys of a variable's distribution. The circle is the median value and the thick vertical line is the standard deviation.

Figure 1 about here, please.

There were 18 observations *with* (UNEXPEC=Y) and 36 *without* (UNEXPEC=N) major unexpected problems. A one-sided Mann-Whitney U test for differences in the median values (6.5 vs 6.0 years) of the two groups gives a p-value of 0.21. This p-value indicates that there is no clear relationship between unexpected problems and length of experience. The power of the Mann-Whitney U test indicates that it is unlikely that we would have found differences in MEXPTOT of less than 2 years. It is, however, more than 70% likely that we would have found a p-value of 0.1 or less if the real difference in mean MEXPTOT of the different UNEXPEC groups had been at least 3 years.

indicates that the least experienced maintainers have an over-proportional large share of the unexpected problems. In fact, all maintenance tasks carried out by maintainers with MEXPTOT less than approximately 2 years faced major unexpected problems! In other words, there seems to be a reduction in the frequency of major unexpected problems from tasks solved by very inexperienced to medium experienced maintainers, but no clear reduction from medium experienced to very experienced maintainers.

H1.2: The median length of application experience (MEXPAPP) is lower on tasks with major unexpected problems (UNEXPEC)

A violin plot of MEXPAPP for each UNEXPEC-category is shown in .

Figure 2 about here, please.

A one-sided Mann-Whitney U test for differences in the median values of the two groups gives a p-value of 0.02. A conservative interpretation of this p-value, taking into account the number of tests conducted, is that we have only a weak indication of a significant relationship. On the other hand, other evidence, e.g., (Jones 1998)

[7] The violin plot is part of the statistical package NCSS.

and the results of H1.3, points in the same direction and supports the presence of a relationship between application experience and maintenance skill. The median MEXPAPP value of UNEXPEC=Y and UNEXPEC=N was 3.0 and 1.25 years, respectively. Cohen's *d* is 0.47, which indicates a medium large size of effect.

This means that there is a significant, medium large, influence from application experience on the degree of unexpected major maintenance problems. Maintainers with more application experience have less maintenance problems than those with less application experience. This indicates that application specific experience is much more important than general maintenance experience to reduce maintenance problems.

H1.3: The proportion of major unexpected problems (UNEXPEC) depends on whether the maintainer participated in the development of the application (DEVSYS and DEVMSYS) or not.

H1.3 is tested, using a chi-square test, comparing the difference between the proportion of tasks with the proportion of major unexpected problems that were:

a) carried out by maintainers who participated in the development of the application changed in the maintenance task (DEVSYS=Y) with the others (DEVSYS=N), see Table 2, and
b) carried out by maintainers who participated in the development of the particular part of the application being changed (DEVMSYS=Y) with the others (DEVMSYS=N), see Table 3.

Table 2 about here, please.

Table 3 about here, please.

A chi-square analysis gives a p-value of 0.07 for the test of independence between DEVSYS and UNEXPEC and 0.08 for the test of independence between DEVMSYS and UNEXPEC. A conservative interpretation of these p-values is that they are not pointing at any significant difference when adjusted for the number of statistical tests. On the other hand, all our tests on the impact of application specific experience on UNEXPEC point in the same direction. For this reason, we interpret the p-values as supporting the indications of a dependency between application experience and degree of unexpected problems. The size of effect (*w*) is medium large, 0.25 and 0.24, respectively.

These results indicate that there is a significant, medium large connection between the participation on the development of the application and degree of maintenance problems. In other words, participation in the initial development of the application is useful to avoid unexpected maintenance problems. This connection may be caused by the usefulness of the knowledge acquired during the design and development of the application. For example, Bratthall et al. (2000) report that information about the design rationale improves the change efficiency. In the studied organisation, the design rationale was *not* included as part of the application documentation, i.e., it was difficult to get access to the design rationale when not participating in the initial application development project.

4.2. Prediction accuracy (H2)

H2.1: The median length of experience (MEXPTOT) is independent of the prediction accuracy category (PRED).

The values of MEXPTOT for the PRED-categories are displayed in .

Figure 3 about here, please.

The median of PRED=OK is 6.0, PRED=OPT 5.5, and PRE=PES 7.0. The Kruskal-Wallis rank test of different median values gives a p-value of 0.5, i.e., the test indicates no significant improvement in prediction performance with increased experience. The power of the test of the difference between the groups with the largest differences of MEXPTOT, i.e., between PRED=PES and PRED=OPT, is relatively low. For example, differences of less than 3 years have a probability of about 60% of achieving a p-value less than 0.1. In other words, differences in median values of, e.g., 1-3 years, may get significant with an increased number of observations. An informal analysis of suggests that the most experienced maintainers (MEXPTOT > 10) are less over-optimistic in their predictions. This observation is, however, based on very few data points and need further studies to be validated.

Consequently, we find no clear connection between the total length of experience and the prediction accuracy. Similarly to the studies described in Section 1 the total length of experience was not a valid indicator of risk prediction ability.

H2.2: The median length of application experience (MEXPAPP) is independent of the prediction accuracy category (PRED)

The values of MEXPAPP for the PRED-categories are displayed in .

Figure 4 about here, please.

The median of PRED=OK is 3.0, PRED=OPT 2.0, and PRE=PES 3.0. The Kruskal-Wallis rank test of different median values gives a p-value of 0.4, i.e., the data indicates no significant improvement in prediction performance with increased experience. The power of the MEXPAPP difference between the group with the largest differences, i.e., between PRED=OK and PRED=OPT, indicates that we should not reject that there is a difference of 1 year or less. It is, however, about 70% likely that we would have found a p-value of 0.1 or less if the difference were at least 2 years.

We find no connection between length of application experience and risk prediction ability. This means that neither the total length of experience nor the length of application specific experience is a valid indicator of risk prediction ability. While more application specific experience led to less maintenance problems (H1.2), it did not lead to better predictions of maintenance problems.

H2.3: The prediction accuracy (PRED) is independent of whether the maintainer has participated in the development of the application (DEVSYS and DEVMSYS) or not.

A chi-square test of independence between DEVSYS and PRED gives a p-value of 0.6 and between DEVMSYS and PRED a p-value of 0.7, i.e., we cannot reject hypothesis H2.3. The data is shown in respectively and . We have found no suitable test of power for chi-square tests, but the low number of observations in some of the table fields should lead to a careful interpretation of these two tests. On the other hand, none of the tests of the hypotheses H2.1 – H2.3 indicate significant relationships between experience and prediction ability, i.e., all tests point in the same direction. This supports that there is, at least, no large improvement of prediction ability connected with the participation in the initial development of the application.

Table 4 about here, please.

Table 5 about here, please.

4.3. A prediction model

Several studies on human judgement indicate that experts tend to focus too much on "representativeness" and neglect "base rates", see for example (Dawes 1988). In our context, this means that the maintainers, when planning a new task, would recall only a few similar maintenance tasks and provide a prediction about unexpected problems based on whether those particular tasks had faced unexpected problems. This strategy may include too few tasks to provide an accurate prediction model. For example, the relation that large tasks have a much higher rate of unexpected problems may not be part of this type of expert judgement based prediction model. For this reason, simple rules may outperform the experts. This section tests the predictions of a simple model relative to the predictions given by the maintainers in our study. The simple model is defined as follows:

IF SIZE < 75 LOC /* small task */
THEN predict "no major unexpected problems"
ELSE predict "major unexpected problems" /* "large" task */

This model can only *indicate* the performance of maintainers relative to the model because compared with the maintainer the model has the advantage of using the actual size of the task. In a fair comparison the model should use the estimated task size as input. This simple model is, however, rather robust with respect to size measurement; an evaluation gives that changing the limit to any value between 50 and 100 LOC would not change the model accuracy very much. The prediction model is developed on the same data on which it is tested. A more proper procedure would be a cross-validation procedure (Bradley and Gong 1983). In our case this would, however, not make any practical difference. We evaluated the use of cross-validation and found no important difference in prediction models or evaluation results.

This simple model was compared with a model based on the predictions of the maintainers:

IF CONFIDENCE = "Y" /* know how to solve the task" */
THEN predict "no major unexpected problems"
ELSE predict "major unexpected problems"

The two models were evaluated using two different criteria:

Criterion 1: Number of incorrect predictions (OPT and PES predictions count equally).
Criterion 1 is meaningful when it is just as important not being too optimistic as being too pessimistic.
Criterion 2: Number of too optimistic predictions (only OPT counts as incorrect prediction).
Criterion 2 is meaningful when only the too optimistic predictions (OPT) are considered harmful. This may, for example, be the case when a low confidence (PES) will lead to more risk analysis and better risk management.

displays the results of the comparison. The field numbers are the numbers of incorrect predictions according to Criterion 1 (#OPT + #PES) and Criterion 2 (#OPT) for the two different prediction models (Simple and Maintainer). indicates that the simple prediction model, based only on the information about the size of the task, is slightly better according to both Criterion 1 and 2! This result is amazing in light of the extensive knowledge the maintainers had about the application and the task. When a one-variable model can result in better predictions, this demonstrates serious shortcomings in how the maintainers learn from previous predictions and/or the process they use to predict the uncertainty of their work. We discuss reasons for this poor prediction performance in Section 6.

Table 6 about here, please.

4.4. Threats to validity

There are several threats to the validity of the results:

Bias in the allocation of maintenance tasks: It is possible that more experienced maintainers get more complex tasks. The interviews with the maintainers indicated that very inexperienced maintainers got the simplest tasks. However, among experienced maintainers the allocations of tasks were probably not very much impacted by the length of experience. Typically, the maintainer with available time and a minimum of competence on the application got the task.

If the more experienced maintainers get more and more complex tasks, our results should be interpreted as regarding the predictability and prediction ability of *own* work. This is, for example, useful information for project leaders. In addition, the lack of learning from experience should be interpreted as a *lower* learning rate than the rate of increase in complexity of tasks.

Lack of realism of prediction process: We do not know whether the maintainers would have given the same predictions and answers if they were questioned by a project or group leader instead of an external researcher. Risk predictions might have been a bit more thorough if the group leader were asking.

Frequently, a maintenance task prediction will be a combination of predictions from more than one expert. There are studies indicating that combined expert predictions are more accurate than predictions based on only one expert prediction (Höst and Wohlin 1998; Fischer and Harvey 1999). Our study is limited to predictions made by one maintainer predicting his/her own work.

Low quality of the data: Although interviews may be better than questionnaires in getting feedback on the inconsistencies in interpretation of important terms (Jørgensen 1995b), it is difficult to ensure a common interpretation of questions such as "were there any major unexpected problems". To some extent, such interpretations will be person dependent. Similarly, there may be low quality answers due to misunderstandings or biases. From the interviews, however, we felt that the questions were well understood and that the interpretations of them were reasonably consistent, maybe due to the rather "homogenous culture" in the maintenance department and the anonymity provided in the study. In fact, we were sometimes surprised by the openness and willingness to spend effort on giving us high quality data.

Lack of meaningful measures: It was difficult to find a good measure on experience; our measure has several limitations. For example, we have not measured experience with different types of programming languages, design methods, testing, etc. This is, perhaps, only a minor limitation. Nearly all the tasks involved understanding and/or changing COBOL, 4GL or C-code; no tasks used any formal design method and all tasks involved some sort of testing.

The threats and limitations of this study should lead to a careful interpretation of the results.

5. RELATED WORK

As described in Section 0, there are studies reporting weak or no correlation between amount of experience and professional judgement. The reasons why the quality of professionals' judgements may not improve much through experience are according to (Brehmer 1980):

- We try to confirm theories, rather than reject incorrect hypotheses.
- The fact that we are able to find a rule is sufficient to believe that we have a valid rule even though we have no experience indicating that the rule is valid. In other words, the confidence in own knowledge increases with the ability to find rules regardless of the validation of these rules.
- In cases where we act on the experience based judgement there will be a number of additional factors that prevent us from detecting that our judgement is incorrect, e.g. self-fulfilling prophesies.
- We tend to prefer deterministic rules even if the relationships between variables are probabilistic.[8] If we find no deterministic rules, we tend to assume that there is no rule at all and start guessing. Applied on software projects this means that we tend to prefer rules where the same software project characteristics lead to the same project outcomes, see also (Kahneman and Tversky 1973). We do this in spite of the fact that a probabilistic rule where similar software projects may have several possible outcomes, each of them with a connected probability, would be more appropriate. The very limited use of probabilistic software effort estimation models in the studied organisation supports that this preference of deterministic rules is the case for maintenance work, too.

A key to improve learning from experience in a probabilistic environment is, according to (Brehmer 1980) and (Hammond 1996), that we are able to detect the probabilistic nature of probabilistic tasks. However, the only way a person can detect this is to evaluate the usefulness of deterministic rules against probabilistic rules. This requires that the person must formulate the hypothesis that the task may be probabilistic and know how to test for this. This knowledge can, according to (Brehmer 1980), hardly be derived from experience alone, but must be taught. Proper education and proper concepts of probability based relations may, thus, be necessary in order to learn from experience in a highly probabilistic environment (Hammond 1996).

There are several software development and maintenance studies of the importance of experience and training, e.g., (Lakhanpal 1993; Subramanian and Zarnich 1996; Taylor, Moynihan et al. 1998). These studies show that experience and training are important. However, they do not describe how different types of experience contribute to maintenance and prediction skills.

The knowledge about one's own prediction abilities is essential. An increase in confidence with the ability to find rules (regardless of the validity of these rules) has been reported by several studies, see for example (Wallsten and Budescu 1983) and (Klayman, Soll et al. 1999). According to these studies there is only a weak positive correlation between how extensive the knowledge is and the confidence level of the individuals, i.e., one should be very careful using the confidence as a predictor of knowledge. The overall tendency is that people are overconfident regarding own judgement ability (Henrion and Fischhoff 1986), (Fischhoff and MacGregor 1986), regardless of experience level. Possible reasons are:

- People seem to believe that they use more information than they actually do when making decisions (Keen 1981; Silverman 1985).
- People have difficulty in searching for weakening information when evaluating their predictions (Einhorn 1978).

Other explanations of overconfidence can be found in (Griffin and Tversky 1992) and (Plous 1993).

Several studies compare the prediction ability of experts with prediction models. Most of these studies, which have been carried out in domains such as clinical judgements and bank loan judgements, see (Dawes 1988) for an overview, indicate that simple linear models are superior to the expert's "global judgement", i.e., intuitive judgement based on the expert's total experience. Dawes (Dawes 1988) explains these findings, amongst others, with the difficulty people have in attending two or more aspects of a situation at once. In particular, this may be difficult when the integration requires knowledge about the distribution of the variables involved. Instead of attending two or more aspects at once people tend to anchor their judgement on an initial prediction and then adjust it in light of the available information. As reported in (Blattberg and Hoch 1990), people tend to have reasonably accurate predictions compared with prediction models if the need for adjustment is small. If, on the other hand, the need for adjustment is large, then very simple prediction models outperform human judgement. The main finding in (Blattberg and Hoch 1990) was, however, that models and people have complementary skills. Models are unbiased; experts can adjust for knowledge impossible to integrate in a model, e.g. very rare occasions. An overview of how to correct expert estimates using statistical methods and how to combine expert and model estimates is given in (Goodwin 2000). Training based on knowledge about the weaknesses of human judgement may significantly improve expert predictions, as indicated in (Fischer and Harvey 1999).

[8] Ashby and Maddox argue that the difference between probabilistic and deterministic rules is unclear and is very much an internal mental process that we have little access to. They also give a summary of the research on when people use deterministic and when they use probabilistic rules to give responses on stimuli (Ashby and Maddox 1998).

There are a few studies that compare expert performance with model performance when predicting effort to develop software, e.g., (Finnie and Wittig 1997; Jørgensen 1997; Myrtveit and Stensrud 1999). These empirical effort estimation studies report, on average, small differences in prediction accuracy between experts and models. In our study we measure estimation inaccuracy as proportion of major unexpected problems, which is different from the accuracy measures based on the deviation between the estimated and the actual effort in those studies. For this reason, our finding that a simple prediction model outperformed the experts does not necessarily contradict those results.

6. DISCUSSION OF THE RESULTS

In this section we try to explain some of the findings described in Section 4 relative to findings in other human judgement studies described in Section 5.

6.1. Unexpected problems

The results indicate a decrease in the proportion of major unexpected problems from tasks carried out by inexperienced to task carried out by more experienced maintainers. This improvement may have been caused by the feedback oriented training of the inexperienced maintainers (see Section 2) and the strong increase in programming and/or application knowledge from 0 to 1-2 years of experience.

Reaching a certain maintenance competence level the feedback was mainly of two types: "syntactic feedback" (e.g. from the compiler on syntactic errors) and "outcome feedback" (e.g., from the test case output or from the users if the task was not solved appropriately). Clearly, when the programming knowledge has reached a certain skill level, the syntactic feedback hardly improves competence. More surprisingly, there are studies indicating that outcome feedback does not enable learning from experience either, see (Balzer, Doherty et al. 1989)**.** According to (Balzer, Doherty et al. 1989), the only way to learn from experience is to get feedback on "task relations", i.e., relations inside a task. In our context this means that it is necessary to get feedback on important maintenance task variables and understand how they relate to each other, e.g., how the properties of the specification are related to the design quality. This type of feedback is typically provided when the maintainer gets a better understanding of the application domain and the structure of the application. As indicated by our results, this is hardly achieved with more general software maintenance experience. Application specific experience is needed and much of the experience from maintaining one application seems difficult to transfer to another application. This finding is in accordance with the emphasis put on the importance of application experience in (Jones 1998).

6.2. Prediction accuracy

We found no improvement in prediction ability with increased experience, and a simple model outperformed the maintainers when predicting major unexpected problems. The prediction ability may thus be low and not improve with increased experience. The related work described in Section 5 gives several reasons for lack of correlation between experience and prediction ability. We believe that most of these reasons are valid for software maintenance predictions. In particular, we believe that:

- The feedback of the predictions was outcome-oriented. It is hard to learn from a comparison of predicted outcome with actual outcome when not having a model of the relationships "inside" the task (cf. the discussion in Section 6.1).
- The prediction approach used by the maintainers was, as far as we observed, mainly based on finding similar completed maintenance tasks and to use the outcome of those cases to predict unexpected problems of the new task, i.e., prediction by analogy. No tools were used to store previous predictions or to support probability (or frequency) based predictions. Neither were there any indications of probabilistic thinking. These factors, we believe, indicate the use of a deterministic model to explain relations important for the predictions. A deterministic model is probably not a proper model for learning from previous predictions on maintenance problems (cf. discussion in Section 5). A means to include a probabilistic model is risk analysis, e.g., as recommended in (Sherer 1997).
- There were, in general, no training opportunities in the maintenance process where the prediction process and the outcome were evaluated. This finding is similar to the lack of training reported in (Taylor, Moynihan et al. 1998).

6.3. Learning from experience with better feedback

To better understand the influence of feedback on the prediction ability we are in the process of conducting a number of follow-up studies, e.g., studies where software professionals predict the effort and risk of maintenance work and get feedback on the prediction accuracy immediately after each prediction. Preliminary results from those studies are as follows:

- While most developers do not improve their risk predictions when provided with prediction accuracy feedback, a few of them improve the risk predictions very soon. Analysing the prediction strategies, described by the participants themselves, has not yet resulted in any evidence that some prediction strategies are better than others in integrating previous experience. In other words, more research is needed to understand which process and/or person characteristics that lead to prediction improvements.
- Many developers do not have proper learning models. For example, we find that some software developers do not include the risk of very unlikely problems in their risk estimates. There are potentially a large number of unlikely problems, and the probability that at least one of them occurs, increases with the length of the task. We found that even when the maintainers observed that something unlikely frequently occurred, they did not integrate very unlikely events in their future risk predictions.
- Maintainers are frequently met with a positive response on high confidence predictions, i.e., on predictions that there will be no problems. For example, we find that maintainers who provide high confidence predictions were interpreted by other software professionals as more knowledgeable than maintainers providing less confidence predictions, even in situations were the low confidence predictions were clearly more accurate! This means that the maintainers may have a strong personal motivation for providing high confidence predictions.

There are several obstacles when learning from experience. More knowledge about these obstacles means that we may be able to remove some of them or, at least, know when we can expect accurate risk predictions.

7. CONCLUSIONS AND FURTHER WORK

Without proper learning situations and feedback, more maintenance experience does not necessarily lead to improved maintenance or prediction abilities. This is illustrated by our finding that the frequency of major unexpected problems did not decrease with increased general software maintenance experience after reaching a moderate experience level. There was no improvement of the accuracy of predictions of own work with increased experience. In fact, a simple prediction model predicted major unexpected problems better than did the maintainers. Our findings are similar to the findings reported in several studies from other domains. Analysing the maintenance process we found several learning from experience hindrances. In particular, we believe that the amount of learning situations and the quality of the feedback should be improved to enable learning from experience.

At present, we study how an improved learning and training process may impact risk prediction skills. In particular, we focus on the timing and the type of feedback needed to improve the learning from experience.

Appendix A

TASK ID.	MEXPTOT	MEXPAPP	DEVSYS	DEVMSYS	UNEXPEC	CONFIDENCE	SIZE (LOC)
1	7	6	Y	N	N	Y	250
2	4	3.5	Y	Y	N	N	4
3	3	2	N	N	N	N	50
4	17	2	Y	Y	N	Y	15
5	10	3	Y	Y	N	N	7
6	7	5	Y	Y	N	N	600
7	22	22	Y	Y	N	N	250
8	8	0.3	N	N	Y	N	1000
9	18	3	Y	N	Y	N	1
10	2.5	2.5	Y	N	N	Y	200
11	9	0.5	Y	Y	Y	Y	200
12	4	1	N	N	Y	N	25
13	1	0.5	Y	N	Y	N	700
14	2	0.3	Y	Y	Y	Y	20
15	3	3	Y	N	Y	Y	600
16	25	3.5	N	N	Y	N	2100
17	10	10	Y	N	Y	Y	29
18	8	5	Y	N	N	N	500
19	5	4.5	Y	N	N	Y	1
20	2.5	2	Y	N	N	N	10
21	3	3	Y	N	N	N	40
22	6	0.5	N	N	Y	N	25
23	7	3	N	N	Y	Y	140
24	8	0.3	N	N	Y	Y	435
25	4	0.5	N	N	N	Y	15
26	7	4.5	Y	N	N	N	11
27	6	6	N	N	N	Y	900
28	3	3	N	N	N	Y	1
29	15	4	Y	N	N	Y	5
30	5	4.5	N	N	N	Y	10
31	9	6	Y	Y	Y	N	15
32	13	0	N	N	N	N	5
33	6	0.5	Y	N	N	Y	15
34	7	7	Y	N	N	Y	30
35	3	0.2	N	N	N	N	170
36	6	0.3	N	N	Y	Y	1600
37	17	17	Y	N	N	Y	100
38	4	3.5	Y	Y	Y	Y	150
39	2	1.5	N	N	Y	Y	0
40	9	7	N	N	N	Y	30
41	10	1.5	Y	Y	N	Y	4
42	15	2	Y	Y	N	N	5
43	3	3	N	Y	N	Y	1
44	16	1	Y	Y	N	Y	350
45	13	4	Y	Y	N	Y	200
46	10	0.3	Y	Y	N	N	118
47	3	0.3	Y	Y	N	Y	10
48	5.5	5.5	Y	Y	N	Y	1
49	5	4.5	Y	Y	N	Y	5

50	4	4	Y	Y	N	Y	3
51	5	3	Y	Y	N	Y	50
52	14	1	Y	Y	N	Y	20
53	1	0.6	N	N	Y	N	100
54	5	2.5	Y	N	Y	Y	3

Acknowledgements

Thanks to Erik Arisholm for his very useful improvement suggestions. We gratefully acknowledge the support from our industrial partners. The research project is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References

1. (1990). *Oxford advanced learner's dictionary*. Oxford, Oxford University Press.

2. Abdel-Hamid, T. K. and S. E. Madnick (1986). Impact of schedule estimation on software project behavior. *IEEE Software* 4: 69-75.

3. Albrecht, A. J. (1979). Measuring application development productivity. *Joint SHARE,*

4. *GUIDE and IBM application development symposium*.

5. Ashby, F. G. and W. T. Maddox (1998). Stimulus Categorization. *Measurement, judgment, and decision making*. M. H. Birnbaum. San Diego, California, Academic Press Limited: 252-293.

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

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

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

9. Boehm, B. (1981). *Software Engineering Economics*. Englewood Cliffs, NJ, Prentice-Hall.

10. Bradley, E. and G. Gong (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistican* 37(1): 36-48.

11. Bratthall, L. L., E. Johansson and B. Regnell (2000). Is a Design Rationale Vital when Predicting Change Impact? - A controlled Experiment on Software Architecture Evolution. *Conference on Product Focused oftware Process Improvement (PROFES'2000)*, Oulu, Finland, Springer-Verlag, Berlin.

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

13. Camerer, C. F. and E. J. Johnson (1991). The process-performance paradox in expert judgment. *Toward a general theory of expertise*. K. A. Ericsson and J. Smith. Cambridge, Cambridge University Press: 195-217.

14. Cohen, J. (1962). The statistical power of abnormal-social psychological research: A review. *Journal of Abnormal and Social Psychology* 65: 145-153.

15. Cohen, J. (1977). *Statistical power analysis for the behavioral sciences (Rev. ed)*. New York, Academic Press.

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

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

18. Doane, S. M., J. W. Pellegrino and R. L. Klatzky (1990). Expertise in a computer operating system: conceptualization and performance. *Human Computer Interaction* 5: 267-304.

19. Einhorn, H. J. (1978). Confidence in judgment: Persistence of the illusion of validity. *Psychol. rev.* 85(5): 395-416.

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

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

22. Ericsson, K. A. and A. C. Lehmann (1996). Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annu. Rev. Pshychol.* 47: 272-305.

23. Finnie, G. R. and G. E. Wittig (1997). A comparison of software effort estimation techniques: using function points with neural networks, case based reasoning and regression models. *J. Systems Software* 39: 281-289.

24. Fischer, I. and N. Harvey (1999). Combining forecasts: What information do judges need to outperform simple average? *International Journal of Forecasting* 15: 227-246.
25. Fischhoff, B. and D. MacGregor (1986). Calibrating databases. *Journal of the American Society of Information Sciences* 37: 222-233.
26. Goodwin, P. (2000). Correct or combine? Mechanically integrating judgmental forecasts with statistical methods. *International Journal of Forecasting* 16(2): 261-275.
27. Griffin, D. and A. Tversky (1992). The weighting of evidence and the determinants of confidence. *Cognitive Psychology* 24: 411-435.
28. Hale, S. P. and D. A. Haworth (1991). Towards a model of programmers' cognitive processes in software maintenance: a structural learning theory approach for debugging. *Software maintenance: research and practice* 3: 85-106.
29. Hammond, K. R. (1996). *Human judgement and social policy*. New York, Oxford University Press.
30. Henrion, M. and B. Fischhoff (1986). Uncertainty assessment in the estimation of physical constants. *American Journal of Physics* 54: 791-798.
31. Holm, S. (1979). A Simple Sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6: 65-70.
32. Höst, M. and C. Wohlin (1998). An experimental study of individual subjective effort estimations and combinations of the estimates. *International conference on software engineering*, Kyoto, Japan.
33. Jones, C. T. (1998). *Estimating software costs*, McGraw-Hill.
34. Jørgensen, M. (1995a). An empirical study of software maintenance tasks. *Journal of Software Maintenance* 7: 27-48.
35. Jørgensen, M. (1995b). The quality of questionnaire based software maintenance studies. *ACM SIGSOFT - Software Engineering Notes* 20(1): 71-73.
36. Jørgensen, M. (1997). An empirical evaluation of the MkII FPA estimation model. *Norwegian Informatics Conference*, Voss, Norway.
37. Jørgensen, M. and D. I. K. Sjøberg (2001). Impact of software effort estimation on software work. *Accepted for publication in Journal of Information and Software Technology*.
38. Kahneman, D. and A. Tversky (1973). On the psychology of prediction. *Psychological Review* 80(4): 237-251.
39. Keen, P. G. W. (1981). Information systems and organizational change. *Comms ACM* 24(1): 24-33.
40. Klayman, J., J. B. Soll, C. González-Vallejo and S. Barlas (1999). Overconfidence: It depends on how, what, and whom you ask. *Organizational behavior and human decision processes* 79(3): 216-247.
41. Lakhanpal, B. (1993). Understanding the factors influencing the performance of software development groups: an exploratory group-level analysis. *Information and Software Technology* 35: 468-173.
42. Lientz, R. J. and E. B. Swanson (1980). *Software maintenance management*. Reading Mass., Addison-Wesley.
43. McKeithen, K. B., J. S. Reitman, H. H. Rueter and S. C. Hirtle (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology* 13(3): 307-25.
44. Miller Jr, R. G. (1981). *Simultaneous statistical inference*. New York, Springer-Verlag.
45. Myrtveit, I. and E. Stensrud (1999). A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering* 25: 510-525.

46. Nosek, J. T. and P. Palvia (1990). Software maintenance management: changes in the last decade. *Journal of Software Maintenance* 2: 157-174.
47. Plous, S. (1993). *The psychology of judgment and decision making*, McGraw-Hill.
48. Rosenberg, W. F. (1996). Dealing with multiplicities in pharmacoepidemiologic studies. *Pharmacoepidemiology and drug safety* 5: 95-100.
49. Sedlmeier, P. and G. Gigerenzer (1989). Do studies of statistical power have an effect on the power of studies? *Psychological Bulletin* 105: 309-316.
50. Sherer, S. A. (1997). Using risk analysis to manage software maintenance. *Software Maintenance: Research and Practice* 9: 345-364.
51. Silverman, B. G. (1985). Software cost and productivity improvements: an analogical view. *IEEE Computer*: 86-96.
52. Soloway, E., B. Adelson and K. Ehrlich, Eds. (1988). *Knowledge and process in the comprehension of computer programs*. The nature of expertice, Lawrence Erlbaum Assoc.
53. Subramanian, G. H. and G. E. Zarnich (1996). An examination of some software development effort and productivity determinants in ICASE tool projects. *Journal of Management Information Systems* 12: 143-160.
54. Symons, C. (1991). *Software sizing and estimating: MkII Function Point Analysis*, J. Wiley and Sons.
55. Taylor, M. J., E. P. Moynihan and A. Laws (1998). Training for software maintenance. *Software maintenance: research and practice* 10: 391-396.
56. von Mayrhauser, A. and M. Vans (1994). Comprehension Processes during Large Scale Maintainance. *International Conference of Software Engineering*, Sorrento.
57. von Mayrhauser, A., M. Vans and S. Lang (1998). Program Comprehension and Enhancement of Software. *IFIP World Computing Congress - Information Technology and Knowledge Engineerin*, Vienna/Budapest.
58. Wallsten, T. and D. Budescu (1983). Encoding subjective probabilities: A psychological and psychometric review. *Management Science* 29: 151-173.
59. Weiser, M. and J. Shertz (1983). Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies* 14: 391-396.
60. Wonnacott, T. H. and R. J. Wonnacott (1990). *Introductory statistics*, John Wiley & Sons.

Biographies:

Magne Jørgensen received the Diplom Ingeneur degree in Wirtschaftswissenschaften from the University of Karlsruhe, Germany, in 1988 and the Dr. Scient. degree in informatics from the University of Oslo, Norway in 1994. He has 10 years industry experience as consultant and manager. He is now an associate professor in software engineering and member of the software engineering research group of Simula Research Laboratory in Oslo, Norway. His research interests include software process improvement, software maintenance, experience databases, software estimation and software measurement.

Dag Sjøberg received an MSc degree in computer science from University of Oslo in 1987 and a PhD degree in computing science from University of Glasgow in 1993. He has five years industry experience as consultant and Group Leader. He is now a Professor in software engineering and is the leader of the research group Industrial System Development in the Department of Informatics, University of Oslo and Head of the Software Engineering group of Simula Research Laboratory. Among his research interests are software evolution, software process improvement, programming environments, object-oriented methods and persistent programming.

Table 1. Definitions of the measures

MEASURE	DEFINITION	SCALE/VALUES
Maintenance Experience in TOTal (MEXPTOT)	Number of years as software developer or maintainer.	Ratio [0,→)
Maintenance Experience on APPlication (MEXPAPP)	Number of years maintaining the application to be changed.	Ratio [0,→)
Application development experience (DEVSYS)	Participation in the development of first or subsequent releases of the application.	Nominal {Y, N} (= {Yes, No})
Module development experience (DEVMSYS)	Participation in the development of the first or subsequent version of the part of application to be changed.	Nominal {Y, N}
Task Solving Confidence (CONFIDENCE)	Confident or not in knowing how to solve the task.	Nominal {Y, N}
Major UNEXPECted problems (UNEXPEC)	Predictability of own work measured as whether there has been at least one major unexpected problem.	Nominal {Y, N}
Prediction Accuracy Category (PRED)	Categories are "Too optimistic" (OPT), "Too pessimistic" (PES) and "correct" (OK). The values are determined as follows: IF (CONFIDENCE = Y AND UNEXPEC = Y) THEN PRED = OPT ELSE IF (CONFIDENCE = N AND UNEXPEC = N) THEN PRED = PES ELSE PRED = OK	Nominal {OPT, PES, OK}
SIZE of task (SIZE)	Sum of lines of code (LOC) added, changed or deleted, including comments.	Ratio [0,→)

Table 2. UNEXPEC vs DEVSYS

	DEVSYS=N	DEVSYS=Y	Total
UNEXPEC=N	9	27	36
UNEXPEC=Y	9	9	18
Total	18	36	54

Table 3. UNEXPEC vs DEVMSYS

	DEVMSYS=N	DEVMSYS=Y	Total
UNEXPEC=N	19	17	36
UNEXPEC=Y	14	4	18
Total	33	21	54

Table 4 PRED vs DEVSYS

	DEVSYS=N	DEVSYS=Y	Total
PRED=OK	11	20	31
PRED=OPT	4	6	10
PRED=PES	3	10	13
Total	18	36	54

Table 5 PRED vs DEVMSYS

	DEVMSYS=N	DEVMSYS=Y	Total
PRED=OK	19	12	31
PRED=OPT	7	3	10
PRED=PES	7	6	13
Total	33	21	54

Table 6 Simple prediction model vs maintainers

MODEL	#OPT + #PES	#OPT
Simple	19	8
Maintainer	23	10

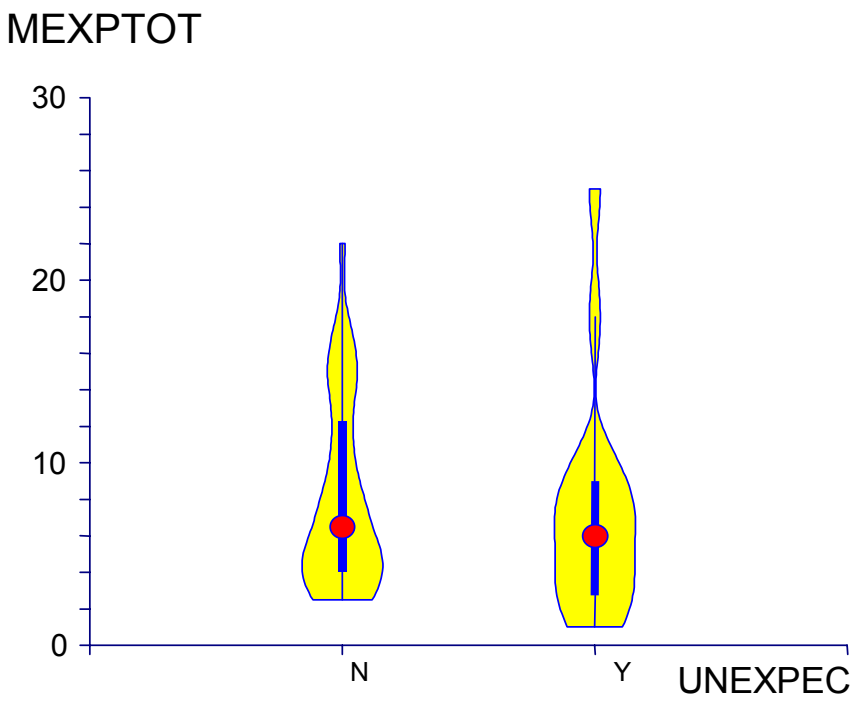

Figure 1. MEXPTOT grouped by UNEXPEC

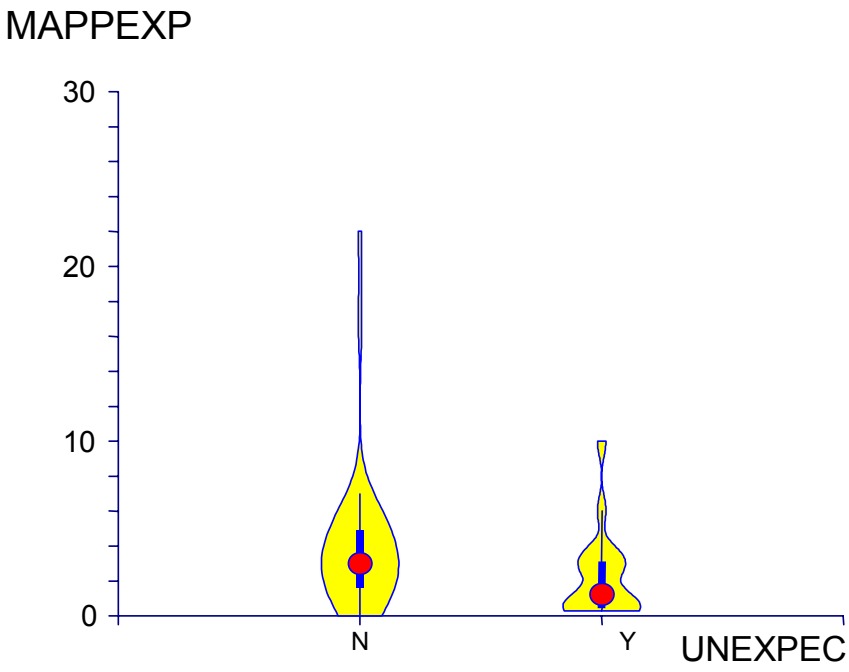

Figure 2 MEXPAPP grouped by UNEXPEC

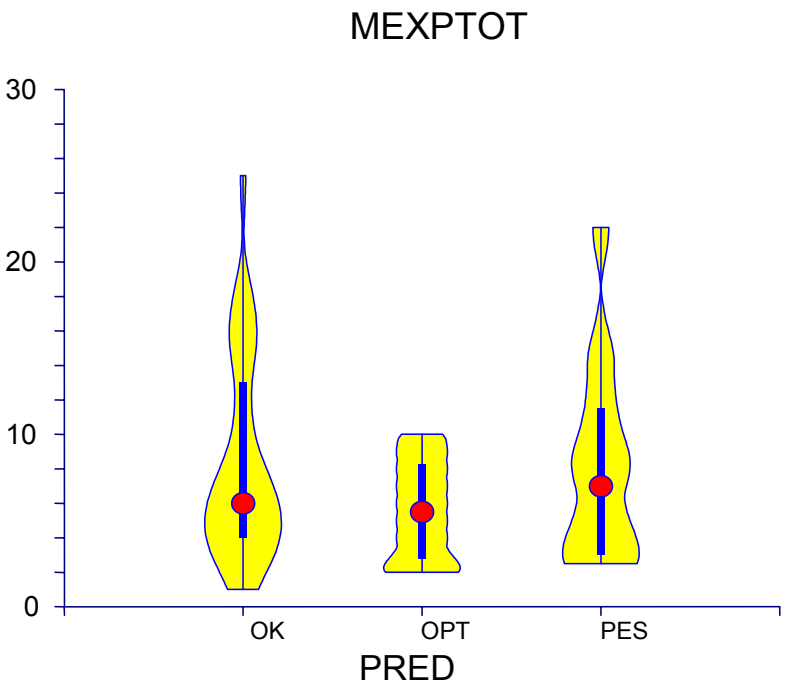

Figure 3 MEXPTOT grouped by PRED

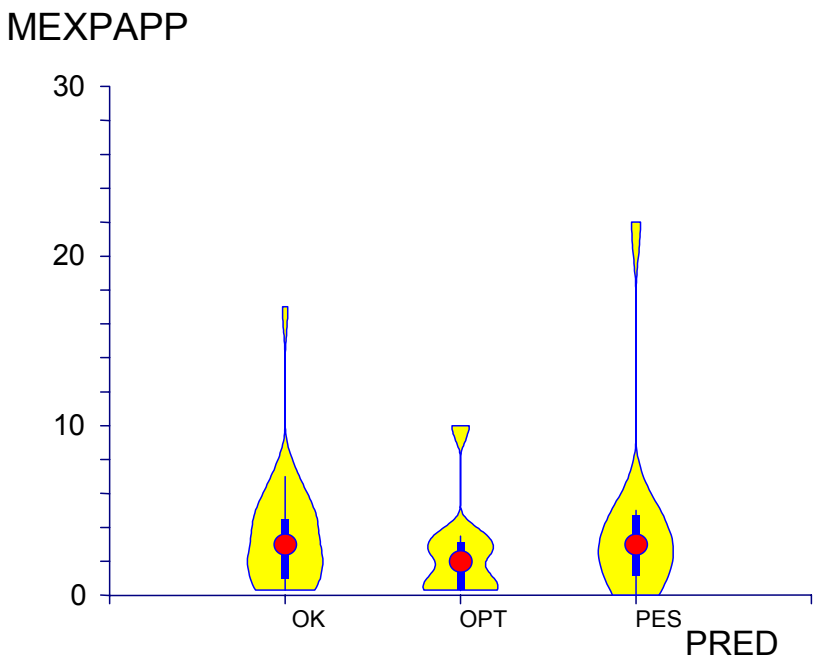

Figure 4 MEXPAPP grouped by PRED