Seaman, C. B. 1996. Communication and organization in software development: An empirical study. Computer Science Department, University of Maryland Technical Report CS-TR-3619.

Yu, E. S., and Mylopoulos, J. 1994. Understanding ‘why’ in software process modeling, analysis, and design. *Proceedings of the 16th International Conference on Software Engineering* Sorrento, Italy.

Report from an Experiment: Impact of Documentation on Maintenance

EIRIK TRYGGESETH eirik@idt.ntnu.no
Department of Computer and Information Science, Norwegian University of Science and Technology, N-7034 Trondheim, Norway

I. INTRODUCTION

Several investigations on software maintenance problems have given high priority to the problems of lacking or inadequate documentation, and a high turnover rate among maintainers (Lientz and Swanson, 1980; Chapin, 1985; Nosek and Palvia, 1990; Foffani, 1992; Dekleva, 1992; Krogstie, 1994).

It is my position that documentation must be actively maintained concurrently with source code to provide new maintainers with a solid basis for quickly becoming productive members of the maintenance team. I define a software system which have all aspects of its functionality described by thoroughly updated documentation to be in *internal equilibrium*.

A part of my doctoral research is to specify a technological framework for how the maintenance process can be supported in order to ensure internal equilibrium of a software system, and how this valuable asset of a system may be used to ensure that members of the maintenance team obtain a high productivity in their work. I will not delve into a long explanation of this, but rather refer to (Tryggeseth, 1997).

I would not try to “validate” my technological research with case studies which can only show the *usability* of a solution, not its *usefulness*. A full validation would involve several pilot projects over a long time period, and an integrated and robust prototype of the technological solution must be built to be used on those pilots. Such an approach is seldom possible in time constrained research such as a PhD study. Rather I propose to use controlled experiments to provide arguments for the research direction chosen. The experiments should include basic aspects of improvements proposed in the technological research so that the soundness of the proposed solution can be discussed by interpolation and generalization.

This paper outlines the results of an experiment organized for this purpose, and lessons learned from carrying it out. The paper is organized as follows: Section II describes the hypotheses tested in the experiment. Section III gives a short overview of the experiment design. Section IV presents the results of the analysis of the experiment data. In Section V, I present some experiences obtained by carrying out the experiment. Finally, Section VI provides some concluding remarks.

II. EXPERIMENTAL HYPOTHESES

In the reminder we refer to maintainers who have only source code available as *category A*. Maintainers who have source code and documentation available are referred to as *category B*.

H1: Maintainers in category B will on the average use *less effort to understand how to fulfill a modification request* than maintainers in category A.

Discussion of H1: When documentation is available, I hypothesize that less effort is used to understand the system, and how to incorporate the changes asked for in the modification request, compared to the case when documentation is not available.

H2: Maintainers in category B will gain *more thorough understanding and provide more detailed specifications* to the solution of the demands given in the modification request than maintainers in category A.

Discussion of H2: Given that H1 holds, maintainers in category B will have a better understanding and more time to specify the pseudo code for the changes needed, and hence be able to specify their proposed changes better. All subjects have equal time available for carrying out the modifications. The faster understanding is acquired, the more time will be available for changes to be made.

H3: The score obtained by a subject in the experiment is expected to correlate positively with the subject's skill. (The experiment scores—M_{pc} and M_u—are defined in Section III.)

Discussion of H3: The reason for including this hypothesis is that it is a common attitude in industry to "hiring the best people money can buy to solve the problems". Is this really true? We would expect subjects skilled in OO and C++ to obtain better results than those with less skills.

III. OVERVIEW OF EXPERIMENT DESIGN

34 subjects participated in the experiment. Their skills in object-orientation and C++ programming were assessed in an initial test. The subjects were partitioned into two categories with similar average and variance based on the skill test score. The following information was available during the experiment:

- Subjects in *category A* had only the source code of the system available. The input and generated output for a small example were also available.
- Subjects in *category B* had the same information as those in category A. In addition the following system documentation were available: The requirements specification, design document, test report, and user manual.

The example system used in the experiment was a program for inserting comments and navigating among them in C++ programs. The system comprised 2.7kSLOC C++ (not commented) and around 100 pages of documentation.

All subjects were presented with the same modification request, calling for several changes in the given software system. The needed changes were mostly enhancements, but existing functions also had to be changed.

A presentation of the system's functionality and a demonstration of the system were given before the modification request was announced. The subjects were to study the available information to decide which changes to make, and then write detailed pseudo code on paper.

The subjects recorded the effort spent on different tasks during the experiment. Two measures were defined to investigate how well the subjects performed in the experiment.

- M_u measures how the subject has understood the system and how to incorporate the changes. Ordinal $\{0, \ldots, 4\}$.
- M_{pc} measures the degree of detail of the pseudo code written to incorporate the changes. Ordinal $\{0, \ldots, 4\}$.

The complete definition of these measures can be found in (Tryggeseth, 1997).

IV. RESULTS FROM DATA ANALYSIS

A. Initial Observations

Figure 1 shows the distribution of effort over the measured variables for the two categories. The label suffices are coded as follows: U=understanding, D=documentation, C=coding. "Time_U_C" means "effort used for understanding the code". The numbers in the figure is the total time in minutes spent on that phase by the subjects in that category. The total time for category B is less than that for category A, since some subjects in category A decided that they were finished before the time limit ran out.

- Category A subjects spent 21.5% more time than category B on trying to understand the system.
- Category B subjects spent 27.5% more time than category A on implementing the changes. The effort saved on code reading can be used for productive work as actually coding the needed changes.
- Most of the time (A: 74%, B: 65%) was spent on system understanding activities. The percentage was lower for category B, as expected.

When documentation was available (category B), subjects spent on the average the same amount of time consulting the documentation as they did with code.

B. Testing of Hypothesis H1

H_0: Time_U(A) = Time_U(B).

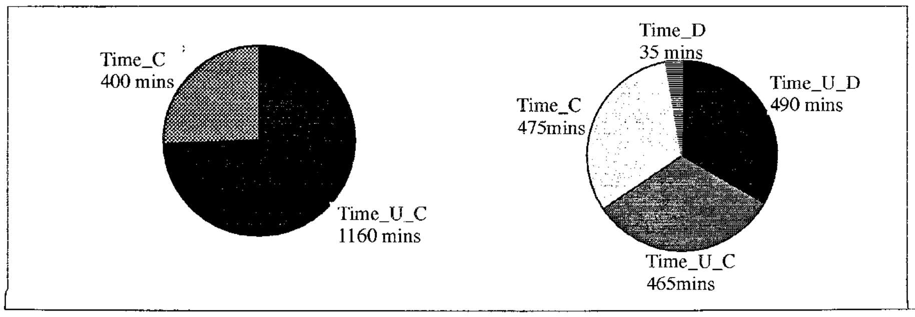

Figure 1. Distribution of effort.

Table 1. Scores on M_{pc} and M_u.

	M_{pc}A	M_{pc}B	M_uA	M_uB
Median	1	2	1	3
Total	19	29	21	32

An independent samples t-test shows statistical significance in the reduction of the Time_U sample mean at a 0.05 level. ($t = 2.14$, critical = 2.06, $p = 0.05$). H_0 is rejected, and H_1 holds under the given conditions.

C. Testing of Hypothesis H2

H_0: The sum of the ranks for M_{pc} and M_u have the same distribution for both category A and B.

The median for the two measures and the sums of their measured values are shown in Table 1.

Since both M_{pc} and M_u are on an ordinal scale, the Mann-Whitney test is used to determine whether H2 holds. Table 2 shows the rank sum (computed using SPSS (Norusis, 1992)) for category A (W). H_0 is rejected for both M_{pc} and M_u, and H2 holds. Using documentation when performing maintenance increase both the level of system understanding and the degree of detail of the changes made during a limited time period.

D. Testing of Hypothesis H3

H_0: There is no significant positive correlation between the subject skills and experimental score.

Table 2. Mann-Whitney results from SPSS 6.1.

	Mean rank				
	A	**B**	**U**	**W**	**2-tailed P**
M_{pc}	10.54	16.46	46	137.0	0.0379
M_u	9.96	17.04	38.5	129.5	0.0139

Table 3. Spearman rank correlation coefficients.

	TR/M_u	**TR/M_{pc}**
Category A	0.2260 ($p = 0.458$)	0,3020 ($p = 0.316$)
Category B	0.8230 ($p = 0.001$)	0.6182 ($p = 0.024$)
All subjects	0.5311 ($p = 0.005$)	0.4309 ($p = 0.028$)

Table 3 shows the computed Spearman rank correlation coefficients between the test results and $M_u(TR/M_u)$, and between the test results and the M_{pc} measure (TR/M_{pc}).

Table 3 shows some very interesting results:

- The correlation coefficients in the first row imply that the null hypothesis for category A cannot be rejected. Weak correlations exist among the test results and experiment scores, but these are small and not significant.
- *However*, for category B, there are strong correlations among the test results and the experiment scores. The null hypothesis *is rejected* for category B, and hypothesis H3 holds at a 0.05 level of significance.

These results deserves some discussion. The fact that hypotheses H1 and H2 hold implies the following:

- The aid of having documentation available during system maintenance reduces the time needed to understand the system and the changes implied by a change request (H1).
- It also enables the maintainer with more time and better knowledge so that he can make more detailed changes to the system given a restricted amount of time (H2).
- The correlations in hypothesis H3 show that the aid of documentation helps the maintainer to use her/his skills better than if no documentation was available. In fact, if a skilled maintainer in category B were not allowed to utilize the aid of the accompanying documentation, he could not expect to do his job better than a person with less skills than himself. On the contrary, when this documentation is available, the skills of the maintainer very much determines the productivity of the maintainer.

This has (at least) two direct implications:

- An organization which is about to employ a maintainer should try to get the best people available. (There is nothing revolutionizing about this.)

- An organization which have hired the best maintainers money can buy, cannot utilize them in an optimal manner if the system they are set to maintain is not documented in a satisfying way. This is at least true in the short run; controlling for this in the long run cannot be done by this experiment design, as the *domain/application knowledge* variable is kept constant at zero level in this experiment.

Preserving the utility of the documentation is therefore important in software maintenance.

V. EXPERIENCES

Initiating an experiment requires a good experiment design. The design of this experiment, including everything from defining the hypotheses and increasing my statistical skills, to preparing the experiment forms to be filled in by the subjects took about three man months. The analysis took about one man month, including writing this paper and a chapter in my thesis. The calendar time used was almost the double. The reason for this was a series of problems in recruiting enough volunteers as experiment subjects. The amount of administrative work with handling the subject recruiting and form management was also considerable.

To summarize, the amount of work needed to finalize the experiment exceeded what I had imagined. However, I believe that the experience obtained from carrying it through is worth the effort.

In retrospect, I encourage every PhD student to include an experiment in their study. This will increase the scientific platform for their continuing career.

VI. CONCLUDING REMARKS

The results of this paper show that documentation is an important tool when maintainers must make changes in a system they are unfamiliar with.

Maintainers who had documentation available used less time to understand how to fulfill a modification request than maintainers who had only source code available. Those who had only source code available used 21.5% more time on system understanding. The difference was statistically significant at a 0.05 level of significance.

Furthermore, the subjects who had documentation available also showed a *better understanding* and a more *detailed solution* to how to incorporate the change, when compared to those who had only source code available. These results also proved to be statistically significant at a 0.05 level of significance.

Finally, the maintainer's skills are not fully exploited when system documentation was not available. For those who had documentation available, there was a significant correlation among the individual skill and the M_{pc} and M_u measures. This was not the case for those who had only source code available. This was confirmed by the subjects in a debriefing after the experiment. The subjects were asked to rank the following statements to express which they thought were most important to improving their experiment results.

1. "I had been allowed to use more time."

2. “The system on which the changes had to be made was smaller.”
3. “I had better knowledge of the C++ language.”
4. “I had more documentation available.”
5. “I had a computer available where changes could be coded.”

The subjects in category A ranked statements 4 and 3 as the most important. Subjects in category B gave highest rank to statements 1 and 2. This shows that subjects in category A had problems in obtaining an overview and understanding of the system. Subjects in category B understood the system and the changes which were needed, but experienced problems in using this knowledge to specify a perfect solution due to the system size.

This experiment shows that documentation is an asset which should be preciously taken care of during software maintenance. Documentation really helps to understand software. When systems gets larger, methodologies and tools will be important to utilize the documentation in efficient ways. This is another aspect of my research which I do not detail here.

Acknowledgments

I am grateful to the BEST programme and the Norwegian Research Council for funding my research. I would like to thank my advisor, associate professor Øystein Nytrø, for valuable comments during the design and analysis of this experiment.

References

Chapin, N. 1985. Software maintenance: A different view. *Proc. of National Computer Conference* 507–513.

CSM92. 1992. Kellner, M. (ed.) *Conference on Software Maintenance* Orlando, Florida, November 9–12, Los Alamitos, California. IEEE Technical Committee on Software Engineering, IEEE Computer Society Press.

CSM NEWSLETTER. 1992. Centre for Software Maintenance Ltd., Association Newsletter. Distributed to members, Centre for Software Maintenance Ltd., Mountjoy Research Centre, Stockton Road, Durham, DH1 3SW, England.

Dekleva, S. 1992. Delphi study of software maintenance problems. *CSM92, 1992* 10–17.

Foffani, F. 1992. Survey on quality in software maintenance in Italy. *CSM NEWSLETTER.*

Krogstie, J. 1994. Survey investigation: Development and maintenance of information systems: Version 1. Technical report, Norwegian Institute of Technology, Department of Computer Science and Telematics, Trondheim.

Lientz, B. P., and Swanson, E. B. 1980. *Software Maintenance Management.* Reading MA: Addison-Wesley.

Norusis, M. J. 1992. *SPSS for Windows Base System User’s Guide Release 5.0.* SPSS Inc.

Nosek, T., and Palvia, P. 1990. Software maintenance management: Changes in the last decade. *Journal of Software Maintenance: Research and Practice* 2: 157–174.

Tryggeseth, E. 1997. Support for understanding in software maintenance. PhD thesis. Dept. of Computer and Information Science, Norwegian University of Science and Technology, ISBN 82-471-0042-8, March 1997.