8-16-1996

An Experimental Investigation Of The Impact Of Individual, Program And Organizational Characteristics On Software Maintenance Effort

Sam Ramanujan
Central Missouri State University, sampath@cmsuvmb.cmsu.edu

An Experimental Investigation Of The Impact Of Individual, Program And Organizational Characteristics On Software Maintenance Effort

Sam RamanujanPh.D.
Central Missouri State University
sampath@cmsuvmb.cmsu.edu

Introduction

During the past twenty years, a number of studies have examined factors affecting software maintenance with the purpose of reducing maintenance expenditures (e.g., Banker et al. 1993, Gremillion 1984). Although many factors affecting these expenditures have been identified (e.g., program size and programmer experience), little progress has been made toward developing a theoretical model that describes the impact of these factors on maintenance activity (Ramanujan and Cooper 1994). The presence of such a model would help organize prior research, reconcile contradictory findings, and provide guidance for future research.

The purpose of this research is to develop a theoretical model of the software maintenance effort and to experimentally evaluate predictions based on the model. The model is based on the Human Information Processing (HIP) literature and can be applied to find ways to reduce maintenance labor and thus maintenance expenditure. Previous research on factors affecting software maintenance is reviewed for validating the application of one of the Human Information Processing models in this context. This grounding in the HIP model is then used to generate propositions for further study. These propositions relate factors such as size, modularity, relevant knowledge, and time pressure to maintenance effort measured in terms of amount of time required to successfully maintain a given program. In addition to developing these propositions, an empirical validation of these propositions will be done in an effort to support the theoretical model.

A Theoretical Model of Software Maintenance

A major component of software maintenance cost is the labor of analysts and programmers responsible for making changes to appropriate programs. This work is largely cognitive in nature and thus it is reasonable to turn to the cognitive psychology literature for guidance in developing a model of software maintenance. According to this literature, psychological complexity of programs and the modifications is the key to understanding the maintenance task.

In the context of software maintenance, psychological complexity refers to the characteristics of software that make it difficult for individuals to understand and modify. For example, a program with many control paths is psychologically more complex than a program with fewer control paths. However, a program with many control paths becomes psychologically less complex as more regularity exists in its branching process (e.g., following a hierarchical structure in branching). The psychological complexity of maintenance can be studied in the context of Atkinson and Shiffrin's (1969) Human Information Processing (HIP) model. This model is chosen because it is simple and also since it is the conceptual core for many other HIP models (e.g., Atkinson and Juola 1974, Shneider and Shiffrin 1977). Ramanujan and Cooper (1994) provide a description of how the HIP model can be used for understanding the maintenance process.

Validation of the Theoretical Model

Overall, there is an agreement between the results of priors studies and the HIP model (see Ramanujan and Cooper (1994) for details). We evaluated the predictions made by the SME model using the results from over twenty research studies and found that only 7 out of 43 SME model's predictions unsupported. The process of forming chunks was observed to be a dominant feature in explaining the relationship between most of the characteristics studied and maintenance effort. This is in accord with Atkinson and Shiffrin (1968) who observed that the processes associated with the STM and buffer are central to HIP model. This

accord of maintenance research with the HIP model supports the descriptive applicability of HIP in this context. Further evaluation of the predictions made by the SME model is conducted in this study in an experimental setting.

Propositions Evaluated in the Study

On the basis of insights gained from applying the HIP model to software maintenance eight propositions were formulated for further evaluation. The first 6 propositions deal with either program characteristics, programmer characteristics, or a combination of the two characteristics. The last two propositions include the effect of organizational characteristics along with the program, request and programmer characteristics. Table 1 maps the factors evaluated and the propositions.

Proposition 1.

The marginal increase in maintenance effort due to the increase in control flow complexity will be greater in larger programs or larger modifications.

Proposition 2.

The marginal decrease in maintenance effort due to higher variable name mnemonicity will be smaller for smaller programs.

Proposition 3.

An increase in semantic knowledge will reduce maintenance effort.

Proposition 4.

The marginal increase in maintenance effort due to the increase in the complexity of control flow will be less for a programmer with higher relevant semantic knowledge.

Proposition 5.

Irrespective of the amount of relevant semantic knowledge, the marginal increase in maintenance effort due to increase in the complexity of control flow will not be significantly different for small programs.

Proposition 6.

If a program has poor documentation and/or logic characteristics, then increased maintenance programmer concentration will decrease maintenance effort considerably. This reduction in maintenance effort will be less if the program has good logic and documentation characteristics.

Proposition 7.

The reduction of maintenance effort due to increased maintenance analyst's/programmer's concentration decreases as the amount of analyst's/programmer's relevant knowledge increases.

Proposition 8.

Greater detail in a repair request will reduce maintenance effort to a greater extent for a novice programmer.

Methodology

An empirical evaluation of the SME model was conducted using a laboratory experiment. The experiment was designed to study the effect of the proposed independent variables and some of their interactions on maintenance effort. The research design associated with the experiment in this study is a variation of the multi-group posttest design with multiple treatments (See Figure 1) (Huck et al., 1974, 274). The treatments include variations in program size, time pressure, request detail, variable name mnemonicity, and control flow complexity. The time taken to correctly maintain a program is the dependent variable and serves as the posttest measure.

Data in this experiment was collected using an automated data gathering tool that simulates the debugging environment for 'C' programs, Program Maintenance Performance Testing System (PROMPTS), designed especially for this research. For each program shown to the subject, PROMPTS records the time required to read the program, read the task and successfully maintain the program. If the subject is unable to maintain the program in a reasonable amount of time (this was determined in a pilot study), the researcher can intervene and allow the subject to proceed to the next maintenance task. The subjects are also provided with two sample maintenance tasks in order to familiarize them with the operation of PROMPTS.

The sample in this study consisted of 100 subjects. One group of fifty subjects came from an introductory "C" language course offered at a large American University whereas a second group of fifty subjects were drawn from the industry, and the average 'C' programming experience for this group was 3 years. An instrument was used to classify the subjects into novices and experts. This instrument had a Cronbach's alpha of .98 and all questions loaded on one factor (accounting for 79% of total variation) suggesting high reliability and validity of the instrument. The low semantic group and the high semantic group had composite scores average of 55.3 and 28.28 respectively. A t-test further confirmed the difference between these two groups.

Results/ Concluding Remarks

Seven of the eight propositions suggested by the model were supported by the results of the experiment. Thus the experiment provides further validation to the HIP model of software maintenance. Propositions 1, 2, 3, 4, 6,7 and 8 were supported at a .05 level of significance, however, the study failed to support proposition 5 (ANOVA/Contrast procedure was used for evaluating each of these propositions.)

The results of the study reveal several interesting relationships between maintenance effort and various programmer, program and organizational characteristics. Documentation characteristics (control flow complexity and variable name mnemonicity) were found to have a significant effect on maintenance effort. This effect was stronger for larger programs. Increase in control flow complexity increased maintenance effort to a greater extent in larger programs when compared to smaller programs. Increase in variable name mnemonicity reduced maintenance effort in larger programs but not for smaller programs.

Programmers with higher level semantic knowledge were found to require less maintenance effort when compared to programmers with lower level semantic knowledge. In addition, the effect of semantic knowledge on maintenance effort was similar for both large and small programs. The results of this study contradict Proposition 5. It was found that even in small programs, the increased control flow complexity resulted in higher maintenance effort for programmers with low level semantic knowledge.

Finally, the results also indicate a positive relationship between organizational characteristics, such as time pressure and repair request detail, on maintenance effort. It was found that increased time pressure (operationalized by having certain programs classified as challenge programs in which a subject can win lottery tickets worth up to $50.00 based on the time taken to successfully modify these programs) decreased maintenance effort while maintaining programs with poor documentation characteristics. This effect was observed to be stronger for programmers with lower level semantic knowledge. It was also

found that increased repair request detail reduced maintenance effort for programmers with lower level semantic knowledge.

This study offers several contributions to MIS practitioners. (For example, it suggests that maintenance effort can be reduced by increasing the programmer's time pressure by setting deadlines or by providing incentives. This practice is especially effective when the programs have poor documentation characteristics or when they are maintained by novice programmers). Even though additional research is necessary to affirm or negate the intensity of influence of the various factors studied, evidence obtained within this research suggests that these factors play an important role in determining the magnitude of maintenance effort. The results of this study are applicable to both traditional and Client-Server environments, however the propositions that deal with semantic knowledge should be re-evaluated since the semantic knowledge instrument is not appropriate for the client-server environment. Finally, we want to emphasize that this study is a small step in the direction toward building cognitive psychology based theory of software maintenance effort that not only predicts "what" factors affect software maintenance effort but also explains "why" and "how" these factors influence software maintenance effort. The HIP model focuses on the individual. However, software maintenance often is a function of team behavior and interaction with clients, especially during the phase in which the programmer tries to understand the client's maintenance requests. Therefore, we also recognize that theories of software maintenance effort can be based on other theoretical perspectives such as sociology and social psychology.

Table 1. Factors Addressed By Propositions

	Semantic Knowledge	***Program Size***	***Program Complexity***	***Program Mnemonicity***	***Program Documentation***	***Request Detail***	***Time Pressure***
Proposition 1	.	x	x	.	.	.	.
Proposition 2	.	x	..	x	.	.	.
Proposition 3	x	.	.	.	.	.	.
Proposition 4	x	.	x	.	.	.	.
Proposition 5	x	x	x	.	.	.	.
Proposition 6	.	.	x	x	x	.	x
Proposition 7	x	.	.	.	.	.	x
Proposition 8	x	.	.	.	.	x	.

R X1 O X2 O X3 O X4 O X5 O X6 O X7 O X8 O X9 O
R Y1 O Y2 O Y3 O Y4 O Y5 O Y6 O Y7 O Y8 O Y9 O

R X1 O X2 O X3 O X4 O X5 O X6 O X7 O X8 O X9 O
R Y1 O Y2 O Y3 O Y4 O Y5 O Y6 O Y7 O Y8 O Y9 O

Legend

R - Random Assignment of Subjects

O - Observation (In the current experiment, an observation is the time taken to maintain a given program)

X1...X9, Y1...Y9 - Treatments

Figure 1 Multi-group Posttest Design with Multiple Treatments

References available upon request from author

Banker, R.D., Datar, S.M.,Kemerer, C.F. and Zweig, D. (1993) Software complexity and software maintenance costs. Communications of the ACM. 36(11), 81-94.

Gremillion, L.L. (1984) Determinants of program repair maintenance requirements. Communications of the ACM. 27(8), 826-832.

Huck, S.W., Cormier, W.H. and Bounds, W.G. (1974) Readings in Statistics and Research. HarperCollins Publishers, New York.

Ramanujan, S. and Cooper, R.B. (1994) A human information processing perspective on software maintenance. Omega. 22(2), 185-203.