Workshop on Empirical Studies of Software Maintenance, Monterey, Nov 1996
Fraunhofer ISBN 3-00-001337-7 p79-83

Strategies for Studying Maintenance[1]

Timothy C. Lethbridge
tcl@csi.uottawa.ca
Department of Computer Science
University of Ottawa, K1N 6N5, Canada

Janice Singer
singer@iit.nrc.ca
Institute for Information Technology
National Research Council, Ottawa, Canada

1. Introduction

In this paper, we present some ideas about how to systematically study the work of software engineers (SEs) who are performing maintenance. In particular, we focus on the questions to which maintenance researchers should seek answers.

1.1 Our research project

Our research project involves applying these ideas to the study of a group of software engineers (SEs) who are evolving a large telecommunications system.

Our goal is to improve the SEs' productivity. The research has two main phases:

Phase 1: Study a significant number of maintainers to thoroughly understand the nature of their work. This paper investigates the questions to ask in such a study. A companion paper [1] discusses some methods by which the questions can be answered.

Phase 2: Develop and evaluate tools to help the maintainers work better. This work will be based on what we learn from phase 1, as well as ongoing involvement with maintainers, As we develop these tools, we plan to use *user-centered design* (e.g. [2]), a process that requires the kind of intimate knowledge of work practices that we are gathering in phase 1.

1.2 Related work

Research into the software maintenance process tends to focus on one or more of the following aspects; it studies ...

- The *organization-wide or group* maintenance process (e.g. [3]). A typical objective is to learn how to improve management practices or methodologies.
- The *products* of maintenance (e.g. [4], [5]); it infers how to improve the process by looking at what was produced by various alternative processes.

[1] This work is supported by NSERC and sponsored by the Consortium for Software Engineering Research (CSER). We would like to thank Jelber Sayyad and Pierre Fauvel for their ideas and feedback.

- The maintenance of *small* programs or systems (e.g. much of the work in program comprehension).

Each of these approaches is important. However, we believe that a much greater emphasis should be placed on a fourth aspect:

- What software engineers actually do when working with large systems; i.e. their *personal maintenance process*.

SEs evolve numerous techniques and strategies through years of experience. By studying the SE's, we can undoubtedly learn much to improve tools, methodologies and processes.

In general, the approaches most similar to ours are those of Boldyreff et al [6] and von Meyrhauser et al [7].

There is promise that others are also thinking in this direction. Industry [8], [9] and the program comprehension community [10] are proposing to make greater use of this approach.

2. Key questions we want to answer about the maintenance process.

We want to learn everything we can about the maintenance process that would have an impact on the kinds of tools we plan to develop. The following are four main categories of questions we are answering as part of our work:

Question 1: What mental models do SEs have of their system and environment?

The tools we develop must correspond to the SEs' mental models, so the tools actively assist them. For example, tools that display or manipulate system information should express that information in a such way that the SE will most readily be able to work with it.

Question 2: What activities are performed by maintainers?

There is no good categorization of all the possible types of maintenance activities at the detailed level. We are creating such an inventory so we can determine the spectrum of tools that might be useful. Most of these activities involve or contribute-to some form of problem solving.

Note that question 1 and question 2 must, by their very nature, be tackled concurrently. By understanding one, we will gain insights into the other.

Question 3. On what scales do maintenance activities vary?

Each maintenance activity can be measured in a variety of ways. One activity might be time consuming and error prone, while another might be mentally demanding, but relatively quick and rewarding. We need to understand how to describe maintenance activities so we can create tools that address different types of difficulties. This question is discussed in section 4.

Question 4: What differences are there among individual maintainers and maintenance environments?

Different SEs will approach any given task in different ways. We therefore need to understand the variability among approaches so we can design tools that are useful to a wide variety of SEs, adapt to individual needs and fit in different environments

3. Answering the questions: The macro- and micro-levels

In general, the questions posed in section 2 can be addressed at the macro-level and the micro-level. The macro-level involves broad concerns whereas the micro-level involves minutae or details. We hypothesize that one must consider both levels (and perhaps various intermediate levels), because they both impact productivity and the kinds of tools useful at each level will be quite different..

Macro-level *mental models* include such things as the SE's view of the architecture of the system. Micro-level mental models include his or her understanding of a particular data item or feature.

Macro-level *individual differences* are the different ways that SEs approach problems. Micro-level individual differences include such things as which specific tools they choose to use.

Macro-level *activities* are those whose goals are of direct concern to the team as a whole, such as fixing a defect or adapting the system to a new environment. On the other hand, the goals of micro-level activities are small parts of the daily work of an individual SE, e.g. searching for a particular data item using 'grep'.

4. Answering the questions: Scales by which to compare maintenance activities

Traditionally, maintenance activities have been primarily compared by measuring the time that maintainers spend performing them. In this section we propose several other important scales that can be used to better understand maintenance activities. Placing an activity on each of these scales will help in the design of tools to assist with that activity.

Scale 1: Amount of time consumed by the activity

This is the most obvious scale; it is included here for completeness. It is well known that a few key activities, such as understanding code, consume a high percentage of an SE's time. In general, we do not know enough about the details of what consumes maintainer's time (in particular the micro-level activities).

Benefits of using this scale:

- We can put more research emphasis and focus tool development on activities that consume more of a maintainer's time.

Hypotheses:

- In most maintenance environments, the amount of time consumed by many activities could be reduced with improved tools or processes.

Most promising study techniques:

- Time-slicing (tools that periodically poll software engineers on what they are doing), think-aloud protocols, observation, automated monitoring.

Scale 2: Cognitive load imposed by the activity

Some maintenance activities are difficult because they tax basic cognitive abilities. The following sub-scales can be identified:

- Memory skills required: The SE often has to keep track of many details while performing some maintenance activity. This can tax maintainers more than SEs developing software from scratch because maintainers will typically be less familiar with any given part of a system.

- Attention-span required: Often the maintainer will have to shift attention many times while changing a system. Keeping oneself focused so as to complete a task can be difficult.

- Level of abstract thinking required: Some activities (e.g. reasoning about the effects of a complex change) involve more abstract reasoning ability than others.

Benefits of using this scale:

- We can put more research emphasis on activities that involve high cognitive load.

Hypotheses:

- High cognitive-load activities are likely to lead to both frustration and errors on the part of maintainers.

- It should be possible to develop tools that reduce the cognitive load of many maintenance activities

Most promising study techniques:

- Think-aloud protocols, time-slicing (with appropriate questions posed), interviewing.

Scale 3: Amount and type of knowledge needed to perform the activity

Whereas the cognitive load scale focused on cognition during problem solving, the knowledge scale focuses on what the user has learned over time or is able to find out easily. Unfortunately, one of the biggest problems organizations face is that there is a high turnover among SEs who perform maintenance, and thus their lack of knowledge can become critical.

Benefits of using this scale:

- We can put research emphasis on tools that help SEs learn better so they can better perform knowledge-intensive activities.

- We can put research emphasis on tools that reduce SEs' need to keep certain knowledge in their memory.

Hypotheses:

- In some cases, it may be possible for a tool to supply the knowledge automatically. In other cases, a tool can help the maintainer by ensuring the knowledge is readily available.

- A judicious combination of improved search and navigation facilities integrated with a simple knowledge base describing the maintained system abstractly should permit SEs with considerably less experience to perform many maintenance tasks.

Most promising study techniques:

- Interviews, think-aloud protocols.

Scale 4: Enjoyability and motivation of the activity

Maintenance has a reputation for being less fulfilling than new development of software; nevertheless, many people enjoy at least some aspects of maintenance. The following are some sources of motivation: Solving complex puzzles, the chance to please users, the feeling of being an expert, the feeling of control due to rapid feedback, job stability and the opportunity to perfect a product.

Benefits of using this scale:

- We want to ensure that SEs are motivated.

- We want to ensure that our tools do not automate-away enjoyable activities and leave only boring ones.

Hypotheses:

- We should be able to develop tools that are enjoyable and rewarding to use.

- Enjoyable and rewarding tools will encourage SEs to be more productive.

Most promising study techniques:

- Interviews, questionnaires, think-aloud protocols, time-slicing (with questions about enjoyment)

5. Conclusions

We are involved in the detailed study of maintainers so that we can build tools to improve their productivity. This paper presents some ideas that can help systematize how to approach such a study.

In particular, a maintenance study should investigate the mental models of maintainers and decompose their activities in significant detail. The study should look at the macro-level and the micro level, and should pose questions about time-consumption, cognitive load, knowledge requirements and enjoyment. Finally, the study should look at how these aspects differ among maintainers and environments.

Additional details about our work can be found at:

http://www.csi.uottawa.ca/~tcl/kbre

References

[1] Singer, J. and Lethbridge, T. (1996), "Methods for Studying Maintenance Activities", *Proc. Workshop on Empirical Studies of Software Maintenance*, Monterey.

[2] Avison D. and Wood-Harper T. (1990), *Multiview methodology.* Oxford: Blackwell Scientific.

[3] Seaman B. and Basili, V. (1996), "The Study of Software maintenance Organizations and Processes", *Proc. Workshop on Empirical Studies of Software Maintenance*, Monterey.

[4] Ohlsson, N., Eriksson, A. and Helander M (1996). "Early Risk-Management by Identification of Fault-prone Modules", *Proc. Workshop on Empirical Studies of Software Maintenance*, Monterey.

[5] Khoshgoftaar, T. and Allen, E. (1996), "Industrial-Strength Software Quality Modeling" ", *Proc. Workshop on Empirical Studies of Software Maintenance*, Monterey.

[6] Boldyreff, C., Burd, E., Hather, R., Mortimer, R., Munro, M., Younger, E. (1995), "The AMES Approach to Application Understanding: A Case Study", In *proc. International Conference on Software Maintenance* (Nice, France).

[7] von Meyrhauser, A., & Vans, A. Marie. (1995), "Program Comprehension During Software Maintenance and Evolution", *IEEE Computer*, 28(8), 44-55.

[8] Foster, J. (1993), "An Industry View on Program Comprehension", *proc. IEEE Second Workshop on Program Comprehension,* Capri.

[9] Peritus Software Services, personal communication (See www.peritus.com for information on a company that has studied software maintainers' personal processes).

[10] Tilley, S. and Smith, D. (1996), *Coming Attractions in Program Understanding*, CMU/SEI-96-TR-019. Software Engineering Institute, Carnegie Mellon University.