What Maintenance Programmers Really Do: An Observational Study

Rebecca Tiarks

University of Bremen
Am Fallturm 1, 28359 Bremen, Germany

beccs@informatik.uni-bremen.de

April 25, 2011

Abstract

Although the field of program comprehension as a research discipline has evolved considerably over the past years, only little is known about how software engineers perform their work. In this paper, we report on an observational study that we have carried out to investigate how software developers understand code when they approach a given maintenance task. We particularly focused on the developers' activities, tools, information needs and their practices. In the study, we observed seven professional programmers at a large supplier in the automotive domain while performing a real maintenance task within their normal workflow. Afterwards we conducted a semi-structured interview to get a deeper insight into the process of program understanding. The focus of our analysis has been on what kind of activities a programmer performs and how those activities depend on each other. We categorize different kinds of activities based on this analysis, highlight challenges faced by the programmers, and discuss the implications of our results on the maintenance process.

1 Introduction

Many tools have been built to help programmers with maintenance tasks and many theories have been proposed to describe how programmers may comprehend systems but only little is known about the activities a developer performs. Changes in software maintenance generally involve three main phases: understanding the existing software, modifying the existing software and revalidating the modified software. This means that a developer must explore the system's source code to find and understand the subset of the code relevant to the change task [2]. As the practices employed by individual programmers when trying to understand a software system are influenced by experience, skill and intuition, there is often a great variability in these practices. Our goal was to understand what kind of activities a developer performs in order to understand a program and which factors associated with these activities influence the comprehension process.

2 Study Method

Our goal was to get deeper insight into the process of program understanding. For this reason we chose an observational study format which was based on the following assumption:

> Developers follow a recurring procedure when performing a software modification task.

To provide a set of of activities we undertook an observational study of professional programmers at a large supplier in the automotive domain. We observed seven programmers working on a real software maintenance task within their normal workflow. To meet the requirement of realism, we needed to study a situation that was representative of a realistic maintenance task [1]. Since observed developers from different departments of the organization, we were confronted with more than one specific software system. All systems were written in C. During the sessions the participants used the tools they would normally use. The participants were asked to select a task in advance that would take them about an hour. At the beginning of each session the participants were asked to describe their task. They were also asked to think aloud while working on the task. After about an hour a semi-structured interview by the author followed. The interview focused on problems and challenges faced by the participants. Each session was audio recorded and field notes recording the time of the events were taken during the session and the interview. Since our study was exploratory, we aimed at make observations based on qualitative data, rather than testing hypotheses using statistical inference.

3 Results

We first investigated the data collected during the observation and afterwards we evaluated the qualitative data gathered during the interviews.

3.1 Observational Study

Our analysis focused on the kinds of activities we observed and on the challenges faced by the programmers. We transcribed the audio recordings produced

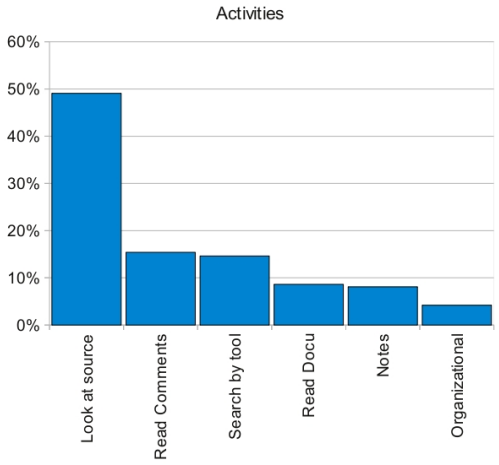

Figure 1: Percentage of time spent on an activity

during the sessions for each subject in combination with the field notes in order to derive abstractions of the actions preformed by the subjects. We categorized the actions in six distinct categories: reading source code, reading comments, reading documentation, searching by tool, making notes, and organizational. The first category summarizes all actions that are associated with reading source code, for instance, reading the source code line by line or searching something by scrolling through the code. Searching for hints in comments is a separate category as we wanted to find out how important comments are for the developer as a source of information. Searching by tool describes all automated searches, either by the IDE used by the developer or an independent tool. Making notes describes the externalization of information by the developer either on a piece of paper or as a digital note. The last category named organizational describes the activities that are not directly related to the task itself but are necessary within the workflow, for instance, checking out a software revision from a version control system or loading a program.

Figure 1 shows the distribution of time for the particular activities. We see that nearly half of the time is spent on browsing through the source code. The subjects spent 15% of their overall time on reading comments and prefer in-line comments as a source of information to written documentation. Automatic searching seems to be an important support mechanism when trying to understand a piece of code. Generally search for information about the system whether through browsing or other tools like *Grep*, figures most prominently. Surprisingly developers spent 8% of their time on the externalization of implicit knowledge.

3.2 Interview

The interview guideline covered several different aspects of program understanding. We wanted to find out how programmers estimate their effort spent on program comprehension. They reported spending about 60%-80% of their time for program understanding. This is due to the special situation that our participants are more concerned with reviewing and assessing software than writing source code themselves. For that reason program understanding makes up the main part of their daily work. During the interviews the developers reported that one of the main difficulties is to create a mental model of the code, which means to find the rationale behind existing code they currently work on. Programmers have a lot of implicit knowledge about the architecture, the dependencies and other details of the code. This knowledge is rarely written down. They reported that in-line comments often describe *what* is done in the code but not *why* and that this information would be very valuable. With regard to documentation and more abstract views of the system such as diagrams, the interviewees had different opinions. Some preferred the code itself and others reported that those kinds of visualization are very helpful. However most of the developers agreed that the code itself is the best source of information about the code. When asked what improvements could be made to support program understanding the common recommendation was to provide the implicit knowledge that is kept in the developers' head to prevent the constant rediscovery of knowledge.

4 Conclusion

Our results show that although there are many measures taken to help programmers understanding the rationale behind the code, e.g. documentation, visualizations and coding guidelines, software development still relies very much on implicit knowledge. When looking for detailed information about code, developers have to switch between multiple tools resulting in loss of parts of their mental model. A solution would be to provide a better interaction between source code and design documents. Furthermore, it is important to capture externalized implicit knowledge to prevent it from getting lost. With respect to our assumption we can state that developers follow their individual recurring procedure but the procedures among all observed developers were highly diverse. Since our observation was restricted to one particular industrial setting it is not clear how this study generalizes to other professional environments. Future work is necessary to get further insight and to understand the generality of these findings.

References

[1] D. I. K. S. et al. Conducting realistic experiments in software engineering. In *ISESE*, pages 17–26, 2002.

[2] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil. An examination of software engineering work practices. In *CASCON '97*, page 21. IBM Press, 1997.