Challenges in Program Comprehension

Rebecca Tiarks
University of Bremen
beccs@tzi.de

Tobias Röhm
Technical University Munich
roehm@in.tum.de

March 8, 2012

Abstract

Program comprehension as a subtask of software maintenance and evolution consumes about half of the time spent by the developers who have to explore a systems' source code to find and understand the subset of the code which is relevant to their current task. The problems encountered during the comprehension process influence the time spent on program comprehension to a great extent. Although many empirical studies have been conducted in the field of program comprehension, only little is known about the challenges developers face when trying to understand a software system. This paper reports on an observational study of 28 professional developers, investigating their behaviour with respect to the occurring problems.

1 Introduction

During software maintenance and evolution, program comprehension is a major subtask, which is driven by the need to change software. Research efforts aiming at addressing challenges of program comprehension can be characterized by both the tools that are used to assist a comprehension task as well as the cognitive theories that provide explanations on how developers understand software. Although research in the field of program comprehension has considerably evolved over the past 20 years and many theories have been proposed to explain how programmers may comprehend software, there are still open issues that require further investigation. A survey of Storey [5] provides an overview of the field including future trends. Still, only little is known about the strategies employed by the developers and the problems they encounter during the comprehension process. Whenever humans are involved, empirical studies such as surveys, case studies or controlled experiments are a rigorous means of empirical research. In the field of program comprehension, a number of studies have been published that involved human subjects. Those studies suffer from limitations calling for a deeper, up-to-date examination. According to [2] much of this work only used small programs or novice users. In order to analyze developers' behavior and problems, we undertook an observational study of 28 professional developers from seven different companies [4]. The goals of the original study were to investigate the strategies and tools a developer uses and the information the developer interacts with. The study showed that the problems faced by the developer during program comprehension are worth being further investigated. Therefore the research question we are exploring in this paper is the following:

- What problems do developers encounter during program comprehension?

With our report, we aim at pointing out some of the problems we experienced in our study to guide future research and tool development.

2 Study Method

The goal of our study was to get a deeper insight into how program comprehension is done in practice. Therefore we chose a combination of observation and interview to approach our goal. We observed developers in their real work environment, and the tasks the participants worked on were chosen by themselves. All participants were professional developers from seven software development companies. We chose participants with different tasks, different project roles, different experiences, and from different company sizes. The observation and the consecutive interview took 1.5 hours in total. Details about the study can be found in the conference paper [4].

3 Problems

During the study, we observed occurring problems and in the interviews, we asked the participants which problems they encounter when trying to understand a software system. That means that the reported problems are not only the problems we noticed during the observation but also the problems explicitly stated by the developers themselves.

Understanding somebody else's code When developers implement a solution for a specific problem, they construct a mapping from a problem domain to a programming domain. Between these domains may exist several intermediate domains and all of these domains have to be reconstructed when trying

to understand the corresponding piece of code. Often this reconstruction is hypothesis-driven. We observed some participants asking and answering questions leading to informal hypotheses which were then verified by the developers. This problem of discovering individual human-oriented concepts is often called *concept assignment problem* [1]. During the interview, many of the participants identified "understanding somebody else's code" as one of the major challenges in program comprehension. One participant argued that "others have another style and other way of thinking", which makes it difficult to reconstruct the mappings mentioned above.

Search and History Several studies have shown that developers quickly forget details when they move to a different location within the source code or even a different task. During the observation, we noticed some developers that wrote down notes either on a piece of paper or used an editor as temporal memory. Some used these notes as a search history and wrote down which variable or function names they had already searched for. Nevertheless some participants searched for the same function or variables names more than once because they did not remember every detail of the result or to validate their hypothesis about that specific piece of code. During the interview, some participants mentioned that they often have to execute the same search many times and that a tool that keeps track of the searches and problem-solving sessions would be extremely helpful. This finding matches with suggestions by Storey [5].

Loosing context because of interrupts, switching between tasks and window changes Discussions, meetings and phone calls are a necessary part of software development. Interrupts disturb developers during their activities. Developers being interrupted lose their current focus and context and it takes them extra time to recover from the interrupts and previously gathered knowledge may be lost. Most of the interrupts we observed were initiated by the developers themselves and were related to knowledge or experience exchange. Another problem we observed were the interrupts caused by window changes or switching between tasks. One participant, for example, had to switch between the development environment and the editor in which he wrote the documentation quite often. After every window change, he had to recover the information needed for the current task.

Finding suitable breakpoints when working with the debugger When trying to locate a bug, many developers used the debugger to inspect the state of the application [3]. Therefore they have to find suitable breakpoints in order to reproduce the data and control flow of the program. We observed several participants that encountered problems to find the *right* breakpoints and during the interview they approved that deciding where to set a breakpoint is "extremely difficult". The strategy they applied was to set a lot of breakpoints in the beginning wherever they felt it could be useful, starting the debugger and diminish the amount by removing those breakpoints that turned out to be irrelevant. That means that they had to inspect a lot of breakpoints to decide whether they were useful or not, which caused significant overhead.

4 Conclusion

The task of understanding code becomes more difficult as software is constantly getting more complex. In this paper, we report on problems developers face when trying to understand a software system. The results reveal interesting aspects of program comprehension that can guide future tool development. We found that one of the main problems is to understand somebody else's code, which is due to the fact that developers implement a solution by applying individual human-oriented concepts. These concepts vary with the type of task, developer personality, the amount of previous knowledge and the type of application, which makes it difficult to reconstruct. A solution would be to represent the implemented concepts in a uniform way. Another problem describes the loss of parts of the mental model. This applies to searches as well as to interrupts. We suggest that tools should implement features to support rediscovery of knowledge by keeping track of the searches and problem-solving sessions. Further as far as interrupts are concerned, a better integration of different tools could help to prevent frequent window changes that distract the developer. With respects to debugging breakpoints we found that developers start the debugging process by inserting a vast number of breakpoints and refine this set during the comprehension process by removing irrelevant breakpoints. To reduce this overhead tools could suggest more appropriate breakpoints according to the context of the current task. Future work is necessary to get further insight and to understand the generality of these findings.

References

[1] T. J. Biggerstaff, B. G. Mitbander, and D. Webster. The concept assignment problem in program understanding. In *Proc. of the 15th ICSE*, pages 482–498, 1993.

[2] J. I. Maletic and H. Kagdi. Expressiveness and effectiveness of program comprehension: Thoughts on future research directions. In *FoSM 2008.*, pages 31–37, Oct. 2008.

[3] G. C. Murphy, M. Kersten, and L. Findlater. How are java software developers using the eclipse ide? *IEEE Software*, 23(4):76–83, 2006.

[4] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej. How do professional developers comprehend software? 2012. accepted for publication.

[5] M.-A. Storey. Theories, tools and research methods in program comprehension: *past, present and future. SQJ*, 14:187–208, 2006.