Component comprehension in context

Ava Heinonen
Department of Computer Science
Aalto University
ava.heinonen@aalto.fi

1. Introduction

Computer programming is a cognitively complex task. It requires the programmer to have a good understanding of the programming language they use and the program they are creating (Pennington, 1987). Many programmers also utilize different third-party components - libraries, packages, and dependencies, as a part of their code. Now, a large number of commercial software solutions contain some functionality achieved using third-party components. In order to use the components correctly, programmers must understand what they do and how they work.

Program comprehension research, or the study of how programmers create mental models of code, has studied programmers' understanding for decades (Storey, 2005). Results acquired with code not containing third-party components indicate that the expertise of the individual programmer (Burkhardt, Détienne, & Wiedenbeck, 2002) and details of the task code is comprehended for (Burkhardt et al., 2002; Kim, Lerch, & Simon, 1995; Détienne, 2002) have an effect on the mental models of the programmer. However, research into programmers' mental models has decreased over the years, and does not yet include comprehension of newer technologies (Bidlake, Aubanel, & Voyer, 2020). Furthermore, research into the comprehension of third-party components has so far been limited to theoretical approaches such as the COTS component comprehension model created by Andrews et al. (Andrews, Stefik, Picone, & Ghosh, 2005). Currently, empirical research into programmers' mental models of third-party components is rare.

2. Aims and Objectives

The aim of this research is to expand the current knowledge of program comprehension to include third-party components and to analyze comprehension in the context of software engineering work. The research aims to assess the cognitive processes and memory structures behind third-party component comprehension extending current cognitive models and providing new insights into program comprehension. Furthermore, the research aims to evaluate how contextual and individual factors known to affect program comprehension affect the comprehension of third-party components. Some of these factors are detailed in Figure 1. With task code we refer to the existing code a third-party component is used as a part of in order to complete a task. With task description we refer to the verbal or written instructions given to a programmer about the details of the task they work on.

The research will provide insight not only to the comprehension of third-party components but also extend the body of knowledge on program comprehension in general. This will also provide new knowledge on how to design technology and auxiliary materials to assist programmers in using third-party components accurately. The central research questions are:

RQ1 How do programmers form mental models of third-party components in professional software engineering settings?

RQ2 What mental model structures of third-party components do programmers display in professional software engineering settings?

RQ3 How does the task context, including task definition and task code, affect the comprehension process and the formed mental models?

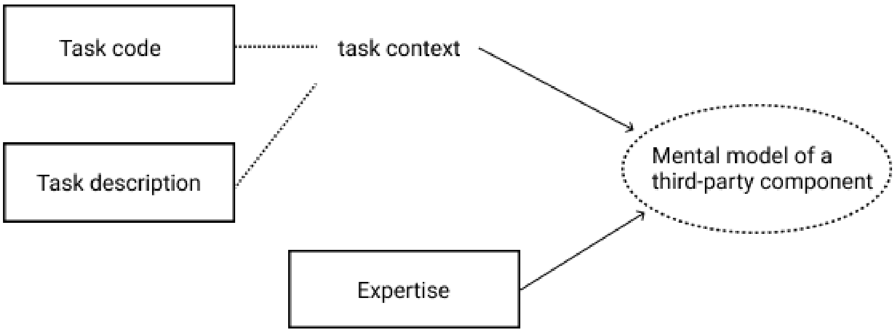

Figure 1 – Factors affecting mental models of third-party components.

RQ4 How does expertise affect the comprehension process and the formed mental models?

RQ5 Can the mental model acquisition process and structure be improved through changes in the third-party components or through auxiliary materials?

3. Proposed research

The research will contain a systematic mapping study and two empirical studies. The research process is depicted in Figure 2. In the first, currently ongoing, phase of this research, we have conducted a systematic mapping study analyzing the different approaches to the structure and acquisition of mental models within the different fields of computer science research and synthesizing existing results of mental model structure and acquisition, as well as results on the effects of context and expertise on mental models. The study will answer research questions 1-4 from the point of view of the existing body of knowledge. The results of the mapping study will provide us with a theoretically backed contextual framework and a study approach to mental models. The results so far not only indicate the lack of research into the comprehension of third-party components, but a lack of research into comprehension in context.

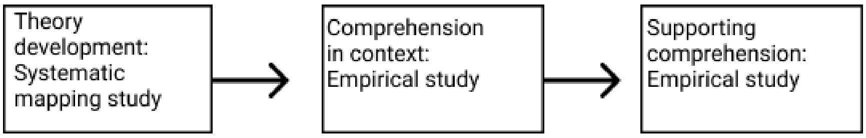
Theory development: Systematic mapping study
Comprehension in context: Empirical study
Supporting comprehension: Empirical study

Figure 2 – Overview of the research process.

In the next steps of this research, we will be conducting empirical research on professional programmers. The proposed research contains two empirical studies: Comprehension in context - using cognitive task analysis to understand the component comprehension process in the context of software engineering work and Supporting comprehension - using results from the first study to identify and test technological tools to assist in comprehension. For the Comprehension in context study, the aim is to map professional programmers third-party component comprehension process in the context of using third-party components as a part of an existing software solution. The study will answer research questions 1-4 with new empirical data. We will be using interviews and observations to analyze the comprehension process. We will also be analyzing programmers' mental models of the existing software and the third-party component. In the second study, we will be utilizing the results of the first study to generate an approach for supporting the comprehension of third-party components, and testing that approach with programmers. This study will concentrate on answering research question 5.

Keywords: Mental Models, Program Comprehension, Third-party Components

4. References

Andrews, A. A., Stefik, A., Picone, N., & Ghosh, S. (2005). A cots component comprehension process. In *13th international workshop on program comprehension (iwpc'05)* (pp. 135–144).

Bidlake, L., Aubanel, E., & Voyer, D. (2020). Systematic literature review of empirical studies on mental representations of programs. *Journal of Systems and Software*, *165*, 110565.

Burkhardt, J.-M., Détienne, F., & Wiedenbeck, S. (2002). Object-oriented program comprehension: Effect of expertise, task and phase. *Empirical Software Engineering*, *7*(2), 115–156.

Détienne, F. (2002). Understanding software: Effects of the task and the textual structure. In *Software design—cognitive aspects* (pp. 105–116). Springer.

Kim, J., Lerch, F. J., & Simon, H. A. (1995). Internal representation and rule development in object-oriented design. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *2*(4), 357–390.

Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, *19*(3), 295–341.

Storey, M.-A. (2005). Theories, methods and tools in program comprehension: past, present and future. In *13th international workshop on program comprehension (iwpc'05)* (pp. 181–191).