Measuring Program Comprehension with fMRI

Janet Siegmund
University of Passau, Germany

1 Introduction

Software development is in essence a human-centered activity, because humans design, implement, and maintain software. Thus, the human factor plays an important role in software engineering. One of the major activities during the entire software-development cycle is program comprehension: Developers spend most of their time with comprehending source code [14]. Thus, if we can support developers in program comprehension, we can reduce time and cost of software development.

To improve program comprehension, for example, by tools or programming languages, we need to measure it reliably—otherwise, we cannot know how or why tools or programming languages affect program comprehension. However, program comprehension is an internal cognitive process that inherently eludes measurement.

2 Measuring Program Comprehension

In a previous paper, we described the state-of-the art measurements techniques that researchers have developed in the past: software measures, subjective rating, measurement of human performance, and think-aloud protocols [4].

First, with **software measures**, such as lines of code or cyclcomatic complexity [5], researchers can make statements about comprehensibility, like "the more lines of code or execution paths source code has, the more difficult it is to understand". However, software measures do not take into account the developer who understands the source code, so their reliability as program-comprehension indicators is limited. Second, using **subjective rating**, researchers show source code to developers and ask them how much they understood of the source code. While this approach takes the developer into account, the perception of the developer is subjective and can easily be biased. Third, instead of using subjective rating, researchers measure **performance of developers** to measure program comprehension. For example, developers are asked to fix a bug, and the time they needed to succeed is taken as comprehensibility indicator; the faster developers provide a correct bug fix, the more easier source code is to understand. However, this does not allow researchers to make any statements about the comprehension process, but only the result of the process. Fourth, with **think-aloud protocols** [13], developers verbalize their thoughts during working with source code. This way, the process can be observed, but can also be biased, in that developers filter their thoughts before saying them out loud.

Thus, all state-of-the-art approaches are fundamentally limited: When evaluating whether a new tool or language improves program comprehension, we can only observe *that* an improvement took place, but not *why* or *how*. While we slowly proceed toward better tools or programming languages, we still have no clear understanding of what happens during program comprehension.

3 Program Comprehension and fMRI

To overcome the limitations of the state-of-the-art approaches, we decided to use *functional magnetic resonance imaging (fMRI)* to measure program comprehension. fMRI has proved successful in cognitive neuroscience to observe cognitive processes, such as language comprehension or problem solving [2]. Since program comprehension as we currently understand it is a similar cognitive process, we decided to conduct a first study to measure program comprehension with fMRI.

fMRI measures changes in the oxygen levels of blood caused by localized brain activity. If a brain region becomes active, its oxygen need increases, which manifests in a higher blood oxygen level of that region, which is measured with an fMRI scanner. Thus, when developers understand source code, the blood oxygen levels of certain regions increase. However, there are also activations that are not specific to program comprehension. For example, in our study, participants should understand source code, which included *seeing* the source code. We needed to exclude activation caused by *visual processing*, because it is not specific to program comprehension. To filter out such irrelevant processes, we designed control tasks that differed only in the absence of the program-comprehension process: Participants should locate syntax errors that did not require understanding the source code (e.g., quotation marks or parentheses that do not match and missing semicolons or identifiers). To observe which brain regions are relevant only for program comprehension, we determined the difference in the brain activation caused between the program-comprehension tasks and the syntax-error-locating tasks.

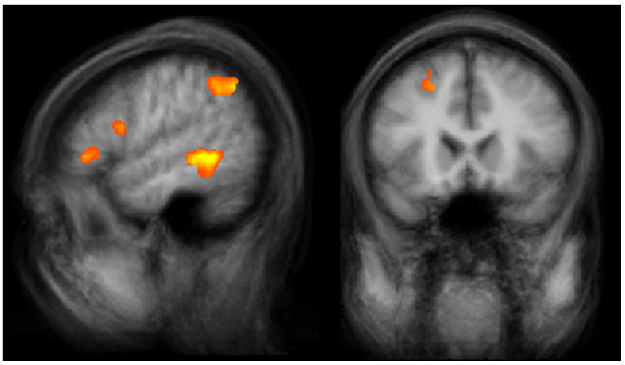

Fig. 1: Activation pattern for program comprehension.

Study Design: We designed several program-comprehension and syntax-error-locating tasks, which we tested in two pilot studies (see project's website `tinyurl.com/ProgramComprehensionAndfMRI/`). For a reliable measurement of the change in the oxygen levels, participants should need between 30 and 120 seconds to complete a task, that is, understand one source-code snippet or find all syntax errors in one snippet. We excluded all tasks for which participants needed less or more time, resulting in 12 program-comprehension and 12 syntax-error-locating tasks (in [10], we present the pilot study in detail).

Results and Discussion: With these tasks, we measured 17 participants with an fMRI scanner [11]. We computed the average over all tasks and all participants. In Figure 1, we show the resulting activation pattern. The highlighted regions are related to language processing, working memory, and problem solving, which is in line with the current understanding of program comprehension. For example, there is a line of fMRI research that observes natural-language and artificial-language processing by letting participants understand natural-language text or artificial words that follow or not follow grammars [1, 7, 12]. Regarding problem solving, researchers observed similar activated brain regions as we did when participants completed Raven's progressive matrices test [9] (i.e., completing rows of figures by adding the correct figure) or the Wisconsin card sorting test [3] (i.e., discovering rules according to which cards need to be sorted) [6, 8].

Thus, we found that fMRI is very promising to give us a detailed understanding of program comprehension. In the long run, we can understand why certain tools or programming languages improve program comprehension. Furthermore, as the measurement equipment will get cheaper in the future, we hope to establish fMRI as common measurement technique in empirical software engineering.

Acknowledgments

Thanks to my colleagues André Brechmann, Sven Apel, Christian Kästner, Chris Parnin, Anja Bethmann, Thomas Leich, and Gunter Saake.

References

[1] J. Bahlmann, R. Schubotz, and A. Friederici. Hierarchical Artificial Grammar Processing Engages Broca's Area. *NeuroImage*, 42(2):525–534, 2008.

[2] J. Belliveau, D. Kennedy, R. McKinstry, B. Buchbinder, R. Weisskoff, M. Cohen, M. Vevea, T. Brady, and B. Rosen. Functional Mapping of the Human Visual Cortex by Magnetic Resonance Imaging. *Science*, 254(5032):716–719, 1991.

[3] E. Berg. A Simple Objective Technique for Measuring Flexibility in Thinking. *Journal of General Psychology*, 39(1):15–22, 1948.

[4] J. Feigenspan, N. Siegmund, and J. Fruth. On the Role of Program Comprehension in Embedded Systems. In *Proc. Workshop Software Reengineering (WSR)*, pages 34–35, 2011. `http://wwwiti.cs.uni-magdeburg.de/iti\_db/publikationen/ps/auto/FeSiFr11.pdf`.

[5] N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, second edition, 1997.

[6] Y. Nagahama, H. Fukuyama, H. Yamauchi, S. Matsuzaki, J. Konish, and H. S. J. Kimura. Cerebral Activation during Performance of a Card Sorting Test. *Brain*, 119(5):1667–1675, 1996.

[7] K. Petersson, V. Folia, and P. Hagoort. What Artificial Grammar Learning Reveals about the Neurobiology of Syntax. *Brain and Language*, 298(1089):199–209, 2012.

[8] V. Prabhakaran, J. Smith, J. Desmond, G. Glover, and J. Gabrieli. Neural Substrates of Fluid Reasoning: An fMRI Study of Neocortical Activation During Performance of the Raven's Progressive Matrices Test. *Cognitive Psychology*, 33(1):43–63, 1996.

[9] J. Raven. Mental Tests Used in Genetic Studies: The Performances of Related Individuals in Tests Mainly Educative and Mainly Reproductive. Master's thesis, University of London, 1936.

[10] J. Siegmund, A. Brechmann, S. Apel, C. Kästner, J. Liebig, T. Leich, and G. Saake. Toward Measuring Program Comprehension with Functional Magnetic Resonance Imaging. In *Proc. Int'l Symposium Foundations of Software Engineering–New Ideas Track (FSE-NIER)*, pages 24:1–24:4. ACM, 2012.

[11] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann. Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In *Proc. Int'l Conf. Software Engineering (ICSE)*, 2014. To appear.

[12] P. Skosnik, F. Mirza, D. Gitelman, T. Parrish, M. Mesulam, and P. Reber. Neural Correlates of Artificial Grammar Learning. *NeuroImage*, 17(3):1306–1314, 2008.

[13] M. Someren, Y. Barnard, and J. Sandberg. *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. Academic Press, 1994.

[14] R. Tiarks. What Programmers Really Do: An Observational Study. *Softwaretechnik-Trends*, 31(2):36–37, 2011.