Toward Measuring Program Comprehension with Functional Magnetic Resonance Imaging

Janet Siegmund[σ]*, André Brechmann[θ], Sven Apel[π], Christian Kästner[ω], Jörg Liebig[π], Thomas Leich[δ], and Gunter Saake[σ]
[σ]University of Magdeburg, Germany [θ]Leibniz Inst. for Neurobiology Magdeburg, Germany
[π]University of Passau, Germany [ω]Philipps University Marburg, Germany
[δ]Metop Research Institute, Magdeburg, Germany

ABSTRACT

Program comprehension is an often evaluated, internal cognitive process. In neuroscience, *functional magnetic resonance imaging (fMRI)* is used to visualize such internal cognitive processes. We propose an experimental design to measure program comprehension based on fMRI. In the long run, we hope to answer questions like *What distinguishes good programmers from bad programmers?* or *What makes a good programmer?*

1. INTRODUCTION

Program comprehension is an important process in software development, because programmers spend most of their time with understanding existing source code [12]. However, program comprehension is an internal cognitive process, which we cannot observe directly [7]. Currently, researchers mostly use indirect measures to assess program comprehension [5]. For example, software measures based on properties of code are used to predict how developers understand source code; or the performance in maintenance tasks is used to estimate how developers understand source code. In some experiments, think-aloud protocols are used to observe the strategies developers use to understand source code. However, to the best of our knowledge, researchers have not yet explored a direct way to observe what is happening inside the brain during program comprehension.

In neuroscience, researchers use *functional magnetic resonance imaging (fMRI)* to observe cognitive processes since 1991 [1]. FMRI is based on measuring differences in oxygen levels of blood flow in the brain. If a region in a brain becomes active, its oxygen need increases. To fulfill that increased need, the amount of oxygenated blood increases, and the amount of deoxygenated blood decreases. Both have different magnetic properties, which are used by fMRI to identify which regions in the brain are activated.

*This author published previous work as Janet Feigenspan.

Today, numerous studies analyze the functional organization of brain areas by varying cognitive tasks and task demands. The results can be used to interpret which brain areas contribute to which cognitive processes. For example, Cabeza and Nyberg describe that the prefrontal cortex is activated in almost all tasks that require high-level cognitive functions, such as memory retrieval [3]. Thus, for a number of cognitive processes, we know which brain regions are responsible. This enables us to draw conclusions about how many resources different cognitive processes require and how different cognitive processes might be related.

With our work, we intend to identify brain areas that are activated during the cognitive processes needed for program comprehension. We do not expect to find one area that is activated during program comprehension, but several areas that reflect the different aspects of programming, such as reading words and working with numbers. Thus, we propose to use fMRI to measure program comprehension. This way, we hope to relate program comprehension to other cognitive processes (e.g., reading comprehension) and get a deeper understanding of how developers understand source code. In the long run, we might be able to answer questions like *What distinguishes good programmers from bad programmers? What distinguishes program comprehension from reading comprehension?* Or *What effects do different tools or programming languages on program comprehension have?*. Here, we present a concept of how such questions can be addressed with fMRI and propose a first experimental setup that is able to identify brain areas that are required during program comprehension.

As feedback, we like to know whether measuring program comprehension with fMRI is interesting for the FSE community. Hypotheses about cognitive processes that are specific for program comprehension and how these could be experimentally tested with fMRI would be valuable for us.

2. PRIOR AND RELATED WORK

In previous work, we showed in a controlled experiment that certain software measures are not reliable indicators for program comprehension and that there is no way around controlled experiments [4].

The single most related paper to our work describes the information programmers need to continue their tasks after interruption [9]. To that end, Parnin and Rugaber describe how different types of memory located in different brain areas affect different programming activities. It is similar to

our work, in that it maps brain regions to programming tasks. However, the authors do not use fMRI, but base their work on previous studies that map brain regions to different types of memory. We are not aware of any results regarding program comprehension based on fMRI.

Furthermore, there is work in the domain of neuroscience to analyze cognitive processes. Most related to ours is work based on reading comprehension (i.e., how participants understand written text). For example, Moss and others analyzed brain regions activated during strategic reading comprehension [8].

3. REQUIREMENTS FOR FMRI STUDIES

The most difficult issue in fMRI studies and most other studies that evaluate cognitive processes is to select suitable material and tasks (that is, source code in our case) and devise control tasks that control for brain activation elicited by processes that are needed for programming, but are not specific for it, such as reading itself. It is imperative that source code and tasks without a doubt lead participants to use the cognitive process that is the target of the evaluation, because otherwise, we cannot be sure what we measure (i.e., ensure construct validity). Furthermore, there are requirements specific to fMRI studies: Source code should be short enough, have appropriate difficulty, there must be a control task, and the complete experiment should not last too long.

First, source code needs to be short enough to fit on a screen that is typically used within a magnetic resonance scanner (from here on referred to as scanner). If source code is too large, participants would have to scroll. However, scrolling would also activate brain regions responsible for motor areas, so the activation caused by understanding would be confounded with the activation caused by scrolling.

Second, source code should be neither too difficult nor too easy to solve. If programs are too difficult, participants might not be able to determine the output correctly. In this case, we cannot be sure whether the understanding process took place or whether participants did something else. Furthermore, participants might require too much or too little time to solve a task. Typically, cognitive fMRI experiments require several repetitions of tasks to ensure sufficient statistical power for data analysis. Between these blocks of programming activity, which should be of comparable length of up to 1–2 minutes, periods of rest or other control conditions are required to allow the amount of oxygenated blood elicited by the programming activity to come back to a resting baseline. If a program is too easy to understand (i.e., within few seconds), we might not be able to observe the activation elicited by the comprehension process, because of insufficient demand on the neural processes. In our studies, we have the opportunity to work with undergraduate students; thus, we adapt the difficulty to their ability.

Third, we need control tasks. Imagine we have suitable programs and tasks, what kind of brain activation would we see? Of course, the activation that is necessary for understanding. However, there is additional activation. First, participants see the source code, so there is an activation in the visual cortex (i.e., the part in the brain responsible for perceiving visual information). Second, participants move their eyes to see the source code, so we have to expect activation in the responsible motor cortex. To deal with these additional activations, we need control tasks that ideally only differ to the processes needed for program comprehension, nothing else. This way, we can compute the *difference* of activation between the control task and understanding task and see activation caused only by understanding.

```
1 public static void main(String[] args) {
2   String word = "Hello";
3   String result = new String();
4
5   for (int j = word.length() - 1; j >= 0; j--)
6     result = result + word.charAt(j);
7
8   System.out.println(result);
9 }
```

Figure 1: Source code for one task.

Fourth, one complete session in a scanner should not last longer than one hour. Inside the scanner, participants have to lie as motionless as possible to avoid motion artifacts. However, after an hour, participants start getting restless and lose attention. To avoid bias, the session duration should not be too long.

4. PILOT STUDIES

In this section, we describe the setting of two pilot studies that we conducted to select suitable source code, experimental tasks, and control tasks. In this stage, it is not necessary to observe participants inside an fMRI scanner, because we can use response time and correctness to evaluate the suitability of source code and tasks. Thus, we conducted this pilot study without fMRI. In Section 5, we describe the current setting of our study with fMRI.

4.1 Finding Suitable Understanding Tasks

As tasks, participants should manually compute the output of source-code snippets. If participants computed an output correctly, they must have understood the snippet. As source code, due to the constraints described in Section 3, we selected 23 different snippets from typical algorithms taught in first-year courses at universities. Algorithms of first-year courses have a suitable difficulty, because we have the opportunity to work with second-year students and our participant should be able to solve as many tasks as possible correctly. To illustrate the nature of the programs, we show an example in Figure 1, in which the correct output is *olleH*.

Note that, in all programs, we obfuscated identifier names to avoid giving participants hints about a program's purpose. For example, the variable `result`, which contains the result, is not named after what it contains (i.e., the reversed string), but the purpose of the variable (i.e., to hold the result of the program). This way, we force participants to use bottom-up comprehension, which means that participants analyze source code statement by statement to determine a program's purpose [10]. In contrast, in top-down comprehension, participants state a general hypothesis about a program's purpose based on their knowledge of a program's domain and based on *beacons* (i.e., information in the code that give hints about a program's purpose) [2, 11]. If participants would use top-down comprehension, we would also observe activation caused by memory retrieval, because information in source code is compared with domain knowledge which is stored in memory. Thus, we focus on bottom-up comprehension, because brain activation is not confounded with memory activation. In the long run, if fMRI proves useful to mea-

```
1 public static void main(String[] ) {
2   String word = "Hello';
3   String result = new String();
4
5   for (int j = word.length() - 1; j >= 0; j--)
6     result = result + word.charAt(j);
7
8   System.out.println{result);
9 }
```

Figure 2: Syntax errors for one task.

sure program comprehension, we shall also consider more sophisticated settings measuring top-down comprehension.

In our first pilot study, we evaluated the suitability of selected source-code snippets. To this end, we analyzed the time participants needed to compute an output of a source-code snippet and the correctness of the determined output.

Our participants were 41 second-year students from a software-engineering course at the University of Passau. They completed a basic programming course, so they were familiar with this kind of programs. The experiment was conducted on a computer in a lab at the University of Passau. To present source code and tasks to participants, we used the tool PROPHET [6]. It shows the source code, the elapsed time, and lets participants enter the output of a method. Furthermore, PROPHET logs data (e.g., time per task, answers) of each participant to support analysis.

To evaluate the difficulty of programs, we looked at the response time and correctness. First, the mean response times for the tasks are between 15 and 316 seconds. Based on these results, we excluded six tasks from further studies with response times above 120 seconds, because participants would require too much time to solve them while in the scanner. Furthermore, we excluded one task with a mean response time below 30 seconds to further reduce the variability of fMRI activation due to differences in task duration. Consequently, we have 16 tasks with mean response times between 37 and 104 seconds. When looking at correctness, the 16 tasks were solved correctly by 29 to 41 participants. On average, the number of correct solutions for a task was 37 (90 %) participants. We want as many participants as possible to find a correct solution, because then we can be sure that a comprehension process took place. Since the percentage of correctness is high enough, we do not exclude any task based on correctness.

4.2 Finding Suitable Control Tasks

Suitable control tasks should ideally only differ to the understanding task in whether comprehension takes place or not. Everything else should be equal. Thus, we use the same programs as for the understanding task. As tasks, we ask participants to identify syntax errors that we introduced to the source code. As an example, we show the source code of Figure 2 with syntax errors that we used for our experiment. The errors are in Line 1 (parameter name is missing), Line 2 (wrong character to terminate String Hello), and Line 8 (curly bracket to pass result). For the remaining 15 tasks, we included similar errors (always 3).

To evaluate suitability of our control tasks, we conducted a second pilot study. As participants, we recruited students from the Philipps University Marburg (4), students from the University of Magdeburg (4), as well as one professional Java programmer. All participants were familiar with Java at least at the level of second-year students. Again, we used PROPHET to show source code to participants and collect response times and answers of participants.

To select suitable control tasks, we looked at the response times of participants, which is between 20 and 120 seconds. Regarding correctness, we found that most participants found at least two syntax errors. Thus, we can be sure that participants worked on the tasks and that the tasks are neither too easy nor too difficult.

Based on the pilot studies, we regard the understanding and control tasks as suitable. The next step is to set up the experiment for the fMRI scanner. In the next section, we describe what a session inside a scanner can look like as a first approach to understand the neural basis of program comprehension.

5. PROGRAM COMPREHENSION BASED ON FMRI

5.1 Experiment Design

To better understand our setting, we introduce the typical setting of fMRI experiments. First, there is a measuring stage of about 10 to 20 minutes, in which the brain of participants is measured regarding size and form. This is necessary to map the measured change in blood flow to the brain region. Then, the actual experiment starts. One typical trial is structured in experiment condition, rest condition, control condition, and rest condition.

The experiment and control conditions are the ones we described in the previous section (i.e., understanding, syntax error). In the rest condition, participants relax or do nothing. This is necessary, to let the level of oxygenated and deoxygenated blood return to a baseline level, and because working inside the scanner is exhausting.

Furthermore, we have to make sure that a session inside the scanner does not last too long (cf. Section 3). However, with our 16 tasks, the experiment would be too long. Hence, we excluded another four tasks. We excluded one task with the shortest and one with the longest response time in the second pilot study. Furthermore, we excluded two tasks that are similar to other tasks.

While lying in the scanner, participants are instructed to determine the output of a method (experiment condition) or find three syntax errors (control condition) and press a button when they are finished. They do not say or enter their solution to avoid motion artifacts, activation in the brain region responsible for producing speech, and to minimize the time inside the scanner, which is a typical setting for fMRI studies. To evaluate whether participants solved a task correctly, we ask them after the scanner session to look again at the source code and state the answer. In the rest condition, participants are instructed to relax. During all conditions, participants are told to move as little as possible to avoid motion artifacts. Furthermore, we use an eye tracker to track eye movement during tasks. To show what it is like for participants to lie inside the scanner, we show a photo of one of our participants in Figure 3 (left). To reduce motion artifacts, the head of participants is fixated.

We arranged the order of code of the experiment and control condition such that most of the code in the experiment condition is presented before code in the control condition.

Figure 3: Left: Photo of participants inside the scanner. Right: Example for activation pattern.

This way, participants do not recognize source code from the control condition, but understand a program bottom up.

Currently, we are running the experiment described in the previous section inside the scanner. Conducting and running such experiments is a long process (e.g., getting a time slot to use the scanner, recruiting participants, analyzing the data), so we have only preliminary results to report. So far, we can confirm that the tasks are actually suitable for our purpose, and that a session is not too long for participants. Furthermore, the experiment is interesting for participants.

To give an impression of the results we might get from our study, we show a typical image of brain activation in Figure 3 from a different study of ours. Highlighted regions indicate an activation, in this case mostly the prefrontal and visual cortex. When our studies are completed, we might obtain an image similar to the one in Figure 3 (right).

6. CONCLUSION & VISION

When finished with the measurements, we hope to have a first impression about which brain regions are activated during program comprehension. Specifically, we expect to see activation in the prefrontal cortex, which is active during higher cognitive tasks (what we believe program comprehension is). Furthermore, we believe that programs that contain loops require more cognitive resources than programs without any loops. Thus, we may observe a stronger activation during understanding programs with loops. We also believe that we will observe more eye movements in source code containing loops (which we observe with an eye tracker). Additionally, we may see activation in regions related to reading comprehension, because participants have to read and understand words. Moreover, we believe that verbal working memory capacity is needed to comprehend source code, because words have to be processed. Thus, we expect activation in the related brain areas (left parietal lobe). To sum up, we believe to observe several activations in different brain regions that are related to activities involved in program comprehension and how that is different from reading comprehension.

In the long run, we might be able to find out what distinguishes a good programmer from a bad programmer. Maybe good programmers have a certain activation pattern that completely differs from bad programmers. For example, fMRI studies of expert and novice golfers showed completely different activation patterns when they were thinking about hitting a golf ball. Expert golfers showed activation in one small, distinguished brain region, whereas novices showed activation in several different brain regions. The reason is that expert golfers have abstracted the knowledge of hitting a ball. With program comprehension, it might be similar, such that experts somewhat abstracted the comprehension process.

Having a deeper understanding of program comprehension, we might be able to better teach programming to students and develop tools and languages that support how humans comprehend source code.

To conclude, we are exploring whether we can use fMRI to better understand program comprehension in an ongoing endeavor. To the best of our knowledge, there is no prior empirical work to measure program comprehension using fMRI. So far, we designed and tested the material we are using. Furthermore, we conducted first sessions inside a scanner that show that our experimental setup is feasible.

As a next step, we will continue our measurements. We hope to find a mapping of these brain regions to other, already evaluated cognitive processes, such as reading comprehension, which might enable us to develop a theory of program comprehension processes based on neuroscience.

Acknowledgments. Thanks to Andreas Fügner for the pictures. Feigenspan's and Saake's work is funded by BMBF project 01IM10002B, Kästner's work partly by ERC grant #203099, and Apel's work by the German Research Foundation (DFG – AP 206/2, AP 206/4, and LE 912/13).

7. REFERENCES

[1] J. Belliveau et al. Functional Mapping of the Human Visual Cortex by Magnetic Resonance Imaging. *Science*, 254(5032):716–719, 1991.

[2] R. Brooks. Using a Behavioral Theory of Program Comprehension in Software Engineering. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 196–201. IEEE CS, 1978.

[3] R. Cabeza and L. Nyberg. Imaging Cognition II: An Empirical Review of 275 PET and fMRI Studies. *Journal of Cognitive Neuroscience*, 12(1):1–47, 2000.

[4] J. Feigenspan, S. Apel, J. Liebig, and C. Kästner. Exploring Software Measures to Assess Program Comprehension. In *Proc. Int'l Symposium Empirical Software Engineering and Measurement (ESEM)*, pages 1–10, 2011.

[5] J. Feigenspan et al. On the Role of Program Comprehension in Embedded Systems. In *Proc. Workshop Software-Reengineering (WSR)*, pages 34–35, 2011.

[6] J. Feigenspan and N. Siegmund. Supporting Comprehension Experiments with Human Subjects. In *Proc. Int'l Conf. Program Comprehension (ICPC)*, pages 244–246. IEEE CS, 2012. Tool demo.

[7] J. Koenemann and S. Robertson. Expert Problem Solving Strategies for Program Comprehension. In *Proc. Conf. Human Factors in Computing Systems (CHI)*, pages 125–130. ACM Press, 1991.

[8] J. Moss et al. The Neural Correlates of Strategic Reading Comprehension: Cognitive Control and Discourse Comprehension. *NeuroImage*, 58(2):675–686, 2011.

[9] C. Parnin and S. Rugaber. Programmer Information Needs after Memory Failure. In *Proc. Int'l Conf. Program Comprehension (ICPC)*, pages 123–132. IEEE CS, 2012.

[10] N. Pennington. Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs. *Cognitive Psychologys*, 19(3):295–341, 1987.

[11] E. Soloway and K. Ehrlich. Empirical Studies of Programming Knowledge. *IEEE Trans. Softw. Eng.*, 10(5):595–609, 1984.

[12] R. Tiarks. What Programmers Really Do: An Observational Study. In *Proc. Workshop Software-Reengineering (WSR)*, pages 36–37, 2011.