Measuring Neural Efficiency of Program Comprehension

Janet Siegmund
University of Passau
Passau, Germany

Norman Peitek
Leibniz Institute for Neurobiology
Magdeburg, Germany

Chris Parnin
NC State University
Raleigh, North Carolina, USA

Sven Apel
University of Passau
Passau, Germany

Johannes Hofmeister
University of Passau
Passau, Germany

Christian Kästner
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Andrew Begel
Microsoft Research
Redmond, Washington, USA

Anja Bethmann
Leibniz Institute for Neurobiology
Magdeburg, Germany

André Brechmann
Leibniz Institute for Neurobiology
Magdeburg, Germany

ABSTRACT

Most modern software programs cannot be understood in their entirety by a single programmer. Instead, programmers must rely on a set of cognitive processes that aid in seeking, filtering, and shaping relevant information for a given programming task. Several theories have been proposed to explain these processes, such as "beacons," for locating relevant code, and "plans," for encoding cognitive models. However, these theories are decades old and lack validation with modern cognitive-neuroscience methods. In this paper, we report on a study using functional magnetic resonance imaging (fMRI) with 11 participants who performed program comprehension tasks. We manipulated experimental conditions related to beacons and layout to isolate specific cognitive processes related to bottom-up comprehension and comprehension based on semantic cues. We found evidence of semantic chunking during bottom-up comprehension and lower activation of brain areas during comprehension based on semantic cues, confirming that beacons ease comprehension.

CCS CONCEPTS

• **Human-centered computing → HCI design and evaluation methods**; **Empirical studies in HCI**;

KEYWORDS

functional magnetic resonance imaging, program comprehension, neural efficiency

ACM Reference format:
Janet Siegmund, Norman Peitek, Chris Parnin, Sven Apel, Johannes Hofmeister, Christian Kästner, Andrew Begel, Anja Bethmann, and André Brechmann. 2017. Measuring Neural Efficiency of Program Comprehension. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17)*, 11 pages.
https://doi.org/10.1145/3106237.3106268

1 INTRODUCTION

During program comprehension, the eyes of programmers glide across a computer screen. In just seconds, they can extract a deep understanding from abstract symbols and text arranged in a source-code file. Expert programmers are especially adept at program comprehension of familiar code—their eyes dance around, finding points of interest, called *beacons* (or semantic cues) that provide hints about a program's purpose, such as method signatures, and common programming idioms. *Top-down comprehension* has been used as an umbrella term to describe cognitive processes related to experience and expectation that guide the understanding of source code [7]. Researchers have also theorized that programmers must use preformed knowledge structures, called *plans*, that represent semantic and syntactical patterns of software [8]. For example, an identifier `bubbleSort` indicates the presence of a sorting algorithm and primes a programmer to expect other elements of the bubble-sort algorithm, such as code related to a swap of array elements.

In contrast, when code lacks familiar semantic cues, programmers must use bottom-up comprehension, a cognitive process that involves obtaining meaning from every individual statement before synthesizing them into a holistic understanding of the entire program [32]. Researchers theorized that programmers must hold these elements in working memory, and through a process called *semantic chunking*, they can abstract these pieces of information into higher order concepts [39]. A previous study by Siegmund and others looked at the process of bottom-up program comprehension with functional magnetic resonance imaging (fMRI) [40], a technique used by to understand brain regions activated by cognitive tasks. In that study, participants have been asked to read short source-code snippets with obfuscated identifier names, such that they could use only a bottom-up approach for comprehension; no cues about the program's purpose were present. The study's authors found a network of brain areas activated that are related to natural-language comprehension, problem solving, and working memory—all processes that fit well to our current understanding of bottom-up program comprehension [40].

In this paper, we contrast bottom-up comprehension with comprehension aided by semantic cues. For many decades, researchers have intensely debated these contrasting theories of program comprehension and the role of concepts such as semantic chunking, plans, and semantic cues without any clear consensus. Some researchers have argued that bottom-up comprehension cannot be avoided because meaning always needs to be extracted from perceptual and syntactical information. Others have argued that programmers actively avoid bottom-up comprehension because it is an inherently tedious process with high cognitive load [44]. Still others have debated the mechanics of how plans and semantic cues are used in comprehension. The goal of this study is to understand how comprehension with semantic cues differs from bottom-up comprehension as cognitive processes in the brain. One possibility is that completely different brain areas can be activated when semantic cues are available, which provides evidence of divergent cognitive processes. Another possibility is that similar brain areas can be activated by both processes; however, the processes differ in neural efficiency. *Neural efficiency* is a phenomenon where lower brain activation indicates that a cognitive process is more efficient and thereby is perceived as easier [30].

To investigate these theories, we address three research questions:

RQ1: Can we replicate the results of the study by Siegmund and others?
Since the original study was the first of its kind in software-engineering research and set methodological standards, it is important to replicate the results. If we succeed, we can confirm the current understanding of bottom-up program comprehension and solidify the role of fMRI as important measurement instrument in software-engineering research.

RQ2: What is the difference between bottom-up program comprehension and comprehension with semantic cues in terms of activation and the brain areas involved?
Bottom-up comprehension is inherently a tedious and time-consuming process and causes high cognitive load. In contrast, comprehension based on semantic cues is very efficient, but requires previous experience and knowledge. The intensity of activated brain areas and the intensity of activation should reflect these differences. Such a result would strengthen our understanding of both bottom-up comprehension and comprehension based on semantic cues.

RQ3: How do layout and beacons in source code influence program comprehension?
Different aspects of source code are believed to influence program comprehension as semantic cues, such as beacons (i.e., identifiers that indicate a program's purpose) or the program layout (e.g., indentation of nested loops). We will test their impact in our study.

In our study, we asked programmers to read variations of similar source-code snippets. Each variation differed in the intended comprehension process, that is, bottom up vs. semantic cues. The semantic cues variations additionally differed regarding beacons (present or not) and layout (pretty-printed or disrupted). We conducted the study in a 3-Tesla fMRI scanner at the Leibniz Institute for Neurobiology in Magdeburg.

Most notably, our results confirm the activated areas of the original study. This represents one important step toward establishing fMRI as standard measurement technique to confirm our understanding of the role of various cognitive processes (e.g., language comprehension, problem solving, working memory) in bottom-up comprehension. We also found that comprehension based on semantic cues required less activation in all areas compared to bottom-up comprehension. This provides neurocognitive evidence for the common belief that comprehension based on semantic cues requires a lower cognitive load than bottom-up comprehension. Beyond this, we could not identify a clear effect of how identifiers and layout of source code influence program comprehension, leaving our third research question as yet unanswered.

In summary, we make the following contributions:

- We replicate the first fMRI study on bottom-up program comprehension, providing more data to better understand this central process in programming.
- We explore which cognitive processes are part of comprehension based on semantic cues.
- We provide evidence to strengthen the role of fMRI as measurement technique for understanding program comprehension.
- We make all material and data available at the project's Web site.[1]

Our long-term research agenda looks toward better understanding program comprehension (and related human activities) in software engineering. We do this by extending current neurocognitive models of program comprehension with new experimental data and analysis. Eventually, this shall lead to better teaching methods (e.g., to train language comprehension, problem solving, and working memory), which shall improve programming education and guide the design of more effective programming tools and practices.

2 FMRI BACKGROUND

fMRI is a relatively new technique for the software engineering research community, so we first give a short introduction to its underlying principles.

When cognitive processes occur (e.g., visual perception), relevant brain areas (e.g., the occipital cortex) activate, increasing the amount of oxygen they need [23]. As the body responds by increasing the amount of oxygenated blood in these areas, the amount of deoxygenated blood decreases. Measuring changes to these levels is called the BOLD (Blood Oxygenation Level Dependent) response [9]. fMRI can detect the difference between the magnetic properties of oxygenated and deoxygenated blood, enabling it to locate activated brain areas. The BOLD response begins at the onset of a task, increases for several seconds until plateauing, and remains high until the person finishes the task. After a few seconds, the ratio of oxygenated and deoxygenated blood returns to its baseline value.

Many brain processes occur all the time, not only those that are related to an experimenter's task. To exclude processes that are unrelated to an experiment's goals, fMRI studies require a *control*

[1]https://github.com/brains-on-code/paper-esec-fse-2017/

Table 1: Code Snippets. Snippets in bold were part of the study by Siegmund and others [40].

Semantic-cues	Bottom-up	Syntax
ArrayAverage	CommonChars	Average
BinaryToDecimal	**CrossSum**	DoubleArray
CrossSum	DoubleArray	**Power**
FirstAboveThreshold	**Factorial**	**ReverseIntArray**
Power	**MaxInArray**	**ReverseWord**
SquareRoot	SumUpToN	**Swap**
ContainsSubstrings		
CountSameCharsAtSamePosition		
CountVowels		
IntertwineTwoWords		
Palindrome		
ReverseWord		

condition. For example, when participants locate structural syntax errors in the same or similar source-code snippets, comparable visual perception also occurs, but not comprehension [40]. We can look at the difference in activation between these two conditions, which shows only the activation that is necessary for program comprehension.

fMRI scanners do not make it easy to present code to participants. To do so, we place a small mirror inside the machine near the participant's eyes, which can show about 20 lines of code. Unfortunately, we prefer not to let participants scroll the text, because that could introduce motion artifacts, inducing a bias in the measurement of the BOLD response.

These methodological constraints (the time course of the BOLD response, the need for a control condition, and physical space limitations; more details can be found in Huettel and others [23]) lead to a highly specialized experiment design, which we explain next.

3 EXPERIMENT DESIGN

Our first research question required us to replicate the experimental setup employed by Siegmund and others [40]. To address our second and third research questions, we extended the setup by including additional code snippets that facilitated program comprehension based on semantic cues. The fMRI session was preceded by a training session, in which participants studied the code snippets including semantic cues to gain familiarity with them. This ensures that participants employ a comprehension process based on semantic cues. Once in the fMRI scanner, participants looked at a series of code snippets. For each code snippet, participants had to determine whether it implemented the same functionality as one of the snippets they looked at in the training session. To evaluate the role of beacons and layout on comprehension based on semantic cues, we created 4 versions of the semantic-cues snippets:

- Beacons and pretty-printed layout [*BY*, *LP*]
- Beacons and disrupted layout [*BY*, *LD*]
- No beacons and pretty-printed layout [*BN*, *LP*]
- No beacons and disrupted layout [*BN*, *LD*]

In the fMRI scanner, participants read snippets of all 4 versions, enabling us to observe how each variation affected the activation intensity and activated brain areas of participants. Based on these data, we draw conclusions on the cognitive processes that occurred during program comprehension. In Figure 1, we give an overview of the experiment-design process and the procedure.

Listing 1: Code snippet with beacons and pretty-printed layout (*BY, LP*)

```
public float arrayAverage(int[] array) {
   int counter = 0;
   int sum = 0;

   while (counter < array.length) {
      sum = sum + array[counter];
      counter = counter + 1;
   }

   float average = sum / (float) counter;
   return average;
}
```

Next, we describe the experimental setup in detail. We include the design of the code snippets, the training session, and the imaging setup used in the fMRI scanner. All material is also available at the project's Web site.

3.1 Experimental Conditions

3.1.1 Controlling Comprehension Strategy. To influence which program comprehension strategy participants used, we introduced several mechanics to control the content and presentation style of code viewed during the experiment.

Bottom-up. For bottom-up comprehension, we need to ensure that participants go through the source code statement by statement. To this end, we use the same methodology as in the original study: We obfuscated identifier names, such that they did not convey the meaning of a variable, but only its usage (e.g., a variable holding the result of an algorithm was named `result`, not `reversedWord`). The snippets that we used for each condition are summarized in Table 1. We made sure the snippets had comparable complexity and length to maintain comparability to the study by Siegmund and others [40].

This condition enables us to address RQ1 by replicating the results from Siegmund and others [40].

Semantic Cues. For comprehension based on semantic cues, we need to ensure that participants can find beacons related to expected code. *Beacons* give hints about a program's purpose and set corresponding expectations [7]. For example, a method named `arrayAverage` implies that the method computes the average of an array of numbers. Prior knowledge on averaging helps programmers to quickly confirm whether the method actually computes the average. In Listing 1, we show a corresponding algorithm. The reader should be able to spot the expected loop with the statements to compute the sum and to increment the counter and, within seconds, can confirm that this method indeed computes the average of an array of numbers.

We also reused some snippets of the study by Siegmund and others for the semantic-cues part of our experiment, but modified them to be more amenable to comprehension based on semantic cues (see Table 1 for the snippets that we used). Specifically, we

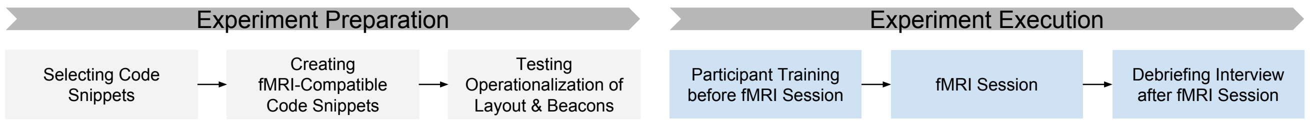
Experiment Preparation
Selecting Code Snippets
Creating fMRI-Compatible Code Snippets
Testing Operationalization of Layout & Beacons
Experiment Execution
Participant Training before fMRI Session
fMRI Session
Debriefing Interview after fMRI Session

Figure 1: Overview of Designing the Snippets and Conducting the fMRI Study

Listing 2: Code snippet with no beacons and disrupted layout (*BN*, *LD*)

```java
public float ayyaoAwyyaky(int[] array) {
    int
          mgqakyy
    = 0;
    int sum = 0;

    while (mgqakyy
          < array.length) {
        sum =
        sum + array[mgqakyy];
        mgqakyy
        = mgqakyy + 1;
    }

    float average
          = sum /
          (float) mgqakyy;
    return
          average;
}
```

added meaningful identifier names according to common Java coding conventions. For example, a variable containing a reversed word is named `reversedWord`.

This condition contrasted with bottom-up comprehension enables us to address RQ2.

3.1.2 Controlling Beacon and Layout Conditions. To further discern which source-code aspects guide comprehension based on semantic cues, we created 4 different versions for each semantic-cues code snippet. To do so, we manipulated two aspects of source code that have been the focus of early studies on semantic-cues comprehension and are believed to have a large impact on the comprehension process: beacons [7] and layout [28]. As beacons constitute a semantic aspect of source code, *layout* constitutes a structural one. In this study, we look at how different program layouts affect the comprehension process. If layout considerably violates common coding conventions, it impairs comprehension based on semantic cues. We show an example in Listing 2 with blank lines and line breaks in unusual locations.

We created a snippet version for each combination of the two aspects, that is: beacons and layout pretty (*BY*, *LP*), beacons and layout disrupted (*BY*, *LD*), no beacons and layout pretty (*BN*, *LP*), and no beacons and layout disrupted (*BN*, *LD*). All versions are available at the project's Web site.

These 4 variants of source code enable us to address RQ3. If beacons and layout do affect comprehension, we should see a difference in the activation pattern between these 4 conditions. If beacons drive the comprehension process, but not layout, the beacon variants should lead to a different activation pattern than the variants without beacons, independent of the layout style. Next, we describe in detail the process of how we created the snippets.

3.2 Designing and Selecting Code Snippets

When using fMRI, it is necessary to compute averages over many conditions [23], so we needed code snippets of similar size and complexity. Furthermore, we had to choose short snippets that fit on the screen inside the fMRI scanner to avoid scrolling. To develop suitable snippets, we followed the procedure established in the original study [40]. First, we conducted several pilot studies, from which we requested feedback from participants, studied response times, and determined correctness. Based on these data, we selected snippets that did lead to our desired 20 to 30 second comprehension time (the BOLD response needs this long for a solid measurement), and did not cause too many incorrect answers. We excluded snippets when participants indicated that they were unsuitable (e.g., requiring mainly visual search).

In the end, we selected 12 snippets that had all similar response times, length, and complexity. The shortest snippet was 8 lines long and the longest had 19 lines. The snippet in Listing 1, which computes the average of an array, was included in our study. The other snippets were similar in size and complexity.

We balanced the content of the snippets, such that 6 snippets manipulated words (e.g., reverse a word), and 6 manipulated numbers (e.g., compute the average, factorial). This way, we ensured that the content of a snippet did not overshadow the activation of comprehension (e.g., in that actually words might result in a certain activation pattern and not the comprehension process itself).

The biggest issue that we faced was to ensure that participants spent 20 to 30 seconds to understand a snippet. Understanding small code snippets with an approach based on semantic cues is very efficient, so snippets like the one in Listing 1 are understandable almost instantaneously, which makes it difficult to measure the BOLD response. Thus, we decided to obfuscate the code through word scrambling.

To find an optimal scrambling degree (i.e., so that we obtain response times between 20 to 30 seconds, but do not bias the comprehension process), we conducted a series of small studies with graduate students in computer science. The methodology is explained on the project's Web site, as it would clutter the experiment description here. In a nutshell, we first experimented with scrambling the code to use Japanese characters, but found that it interfered too much with program comprehension. Next, we tried Caesar shifting (a cipher in which each letter is replaced by another in a static scheme), which was more suitable to elicit comprehension based on semantic cues, but participants adapted to the shifting (e.g., they learned that `qom` means `int`). Thus, we used a variable

Caesar shifting, such that each snippet was created with a different scrambling scheme. Furthermore, we decided to not scramble keywords and calls to library functions, to ensure that each code snippet remained compilable.

Having found a suitable scrambling method, we evaluated the operationalization of the independent variables. To this end, we conducted two pilot studies with undergraduate computer-science students at NC State University.

Testing Beacons. We conducted our first pilot study with 81 students who were all junior and senior undergraduate students in a software-engineering course. To ensure that participants can use comprehension based on semantic cues for the snippets, we trained them before the study. To this end, participants viewed, recalled, and reviewed the snippets to establish familiarity with them. Subsequently, we split the students into two groups: one worked with beacons (*BY*) and the other without (*BN*). The participants had to determine whether a code snippet fulfilled the same function as a snippet from training as correctly and quickly as possible. At the end of the study, students described their cognitive processes in an online questionnaire (see the project's Web site). They stated that they indeed used comprehension based on semantic cues independent of whether beacons were present or not. Their response times also fit into our target interval of 20 to 30 seconds.

Testing Layout. To evaluate the operationalization of layout, we conducted a second pilot study with 12 students who did not participate in the first pilot. The overall design of the pilot study was the same (train, understand, reflect), except that the tested snippets had different layout with beacons present or not. The response times and correctness were similar to the first pilot study and were suitable for our purpose. The students reported that the cognition process is challenged differently, but stayed within the understanding of comprehension based on semantic cues. Thus, also the operationalization of layout proved suitable for our case.

3.3 Participants of the fMRI Study

We recruited 11 programmers from Otto von Guericke University Magdeburg by posting on online and local bulletin boards. In addition to fulfilling the prerequisites for fMRI studies (e.g., no metallic implants), participants needed to have basic knowledge in programming and algorithms. Most participants were familiar with Java or C, at least, at a medium level (answer on a 3-point scale). Only one participant was rather inexperienced with both, Java and C, but had 16 years of Python experience. We designed the snippets to have minimal Java-specific features, and we clarified any questions of participants during the training sessions.

The sample consisted of 5 computer-science students, 3 mathematics students, and 3 professional programmers; 2 participants were female, and 9 were male. All participants were right-handed. The participants' mean age was 25.3 ± 3.82 years. The average programming experience of participants was 1.699 ± 0.385 years, indicating a medium level of programming experience compared with typical computer science students [41].

3 participants agreed to a second session, which results in 14 measured fMRI sessions overall. Participants received 20 Euros compensation per session.

3.4 Task Design

To replicate the study by Siegmund and others (RQ1), we used their tasks: Participants should determine the output of code snippets with given input values. Here, we also made sure to use easily computable results (e.g., factorial of 3), to focus participants on comprehending the program, not computing the output. As control condition, participants also located syntax errors in code snippets, which they had not seen before. None of the syntax errors required participants to understand the code (e.g., they were missing semicolons or closing brackets).

To enable participants to use comprehension based on semantic cues (RQ2 and RQ3), we familiarized them with the semantic-cues snippets and required programming concepts in a training session before they entered the fMRI scanner. Each code snippet had a canonical representation, as would occur in a typical program (e.g., with beacons, pretty-printed layout, syntax highlighting, not scrambled). In training, participants viewed a snippet, recalled it, and then reviewed it again. This was repeated for each snippet.

In the scanner, participants had to decide whether a snippet correctly implemented the same functionality as the snippets seen during training. This task involved recognizing a snippet's purpose and evaluating whether it was working as intended. Participants had to decide as quickly and correctly as possible, with a time limit of 30 seconds for each snippet.

3.5 Experiment Execution

We conducted the experiment at the Leibniz Institute for Neurobiology in Magdeburg. When the participants arrived, they gave their informed consent to participate in the study. Then, they completed a questionnaire on their programming experience [41] and went through the view-recall-review training of correct code snippets. In total, the participants went through 12 trials, in randomized order for each participant. One trial consisted of several tasks:

- Semantic-cues comprehension [*BY*, *LP*, 30 sec.]
- Rest [30 sec.]
- Semantic-cues comprehension [*BY*, *LD*, 30 sec.]
- Rest [30 sec.]
- Semantic-cues comprehension [*BN*, *LP*, 30 sec.]
- Rest [30 sec.]
- Semantic-cues comprehension [*BN*, *LD*, 30 sec.]
- Rest [30 sec.]
- Finding syntax errors or bottom-up comprehension [30 sec.]
- Rest [30 sec.]

From past experience, we know that participants have difficulty lying motionless for more than 30 minutes while concentrating on their tasks. The longer they stay in the scanner, the more restless they get, which causes motion artifacts in the signal. So, we split the 12 trials into 2 sessions, each preceded by the view-recall-review training outside the scanner. When the participants entered the fMRI scanner, they spent their first 6 minutes conducting a pre-trial set of measurements. Then, they looked at 4 semantic-cues comprehension tasks (everyone used the same order) followed by alternating bottom-up and syntax-error discovery tasks. Participants could react to each task by pressing one of two buttons that they held in their hand. In each condition, we asked the participants to be as fast and correct as possible. After each trial, the participants rested to allow their BOLD response to return to their baseline.

Once the participants exited the fMRI scanner, we asked them to explain to us how they solved the tasks. This gave us valuable insights into their comprehension strategies, helping us interpret the results.

3.6 Imaging Methods

Scanner Setting. We carried out the imaging sessions on a 3-Tesla scanner (Philips Achieva dStream, Best, The Netherlands) equipped with a 32-channel head coil. The heads of participants were fixed with a cushion with attached ear muffs containing fMRI-compatible headphones (MR Confon GmbH, Magdeburg, Germany). Participants wore earplugs to further reduce scanner noise by 40 to 60 dB. We obtained a T1-weighted anatomical 3D data set with 1 mm isotropic resolution of the participant's brain before the functional measurement. To capture a whole-head fMRI, we acquired 930 functional volumes in 31 min using a continuous EPI sequence (echo time [TE]: 30 ms; repetition time [TR]: 2000 ms; flip angle, 90°; matrix size, 80 x 80; field of view, 24 cm x 24 cm; 35 slices of 3 mm thickness with 0.3 mm gaps).

Data Preparation. To process the data, we used BrainVoyager™QX 2.8.4 (www.brainvoyager.com). We preprocessed the functional data with 3D motion correction, slice-scan-time correction, and temporal filtering. The anatomical scan of each participant was transformed into the standard Talairach brain [46] (to account for anatomical differences between participants' actual brains). Before group analysis, we spatially smoothed the functional data of each participant with a Gaussian filter (FWHM=4 mm). Based on this transformation and smoothing, we could look at the average activation over all participants.

Analysis Procedure. We conducted a random-effects GLM analysis, defining one predictor for each of the comprehension tasks, and one for the syntax task. To replicate the study by Siegmund and others (RQ1), we calculated the contrast between bottom-up comprehension and the syntax condition using the brain areas defined in that study. To test for differences between bottom-up and semantic-cues comprehension (RQ2), we calculated the balanced contrast between the bottom-up condition and the 4 semantic-cues conditions. To do this, we chose the same level of significance as in the original study (i.e., $p<0.01$, FDR corrected). The resulting clusters of voxels were defined as volumes of interest (VOI) and attributed to their respective Brodmann areas (BA)[2] by using the Talairach daemon (www.talairach.org). Then, we extracted the beta values of the GLM for each participant and condition to identify differences in activation for each of the program comprehension conditions within the defined VOIs.

4 RESULTS AND DISCUSSION

In this section, we present and discuss the results of our study, structured along the research questions.

RQ1: Can We Replicate the Results of the Study by Siegmund and others?

[2]Brodmann areas serve as an anatomical classification system, such that the entire brain is split into several areas on the basis of cytoarchitectonic differences suggested to serve different functional brain processes [6].

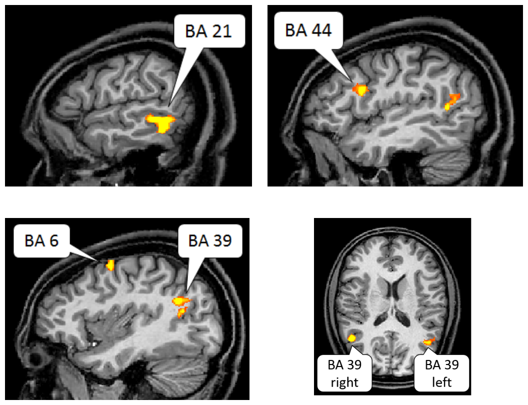

Figure 2: Brodmann-area activation during comprehension based on semantic cues.

Data. The region of interest analysis of the BAs 6, 21, 40, 44, and 47, as defined in the original study, revealed a significantly stronger activation for bottom-up comprehension as compared with syntax-error finding in BAs 21 ($p<0.01$), 40 ($p<0.015$), and 44 ($p<0.01$). For BAs 6 and 47, the contrast was not significant.

Discussion. The result of the activation in BAs 21, 40, and 44 within the brain's left hemisphere confirm the results of Siegmund and others [40]. BAs 6 and 47 were not activated, which may be due to individual anatomical differences between the participant groups or the reduced statistical power of the current experiment (we only had 3 blocks of bottom-up and syntax conditions, compared to 12 in the original study). In the original study, there were only small activation differences in BA 47 and the region sizes of BA 6. Nonetheless, we confirm that the activation of BAs 21 and 44, which are closely related to natural- and artificial-language processing [2, 33, 43], are both integral to comprehend source code. The role of BA 21 lies in processing semantic information at the word level [1, 4, 14], which takes place when participants extract the meaning of single program tokens. BA 44 is believed to play an important role in building syntactic structures [16, 18, 19]. Together with BA 40, they all help to integrate relevant semantic information [16, 22].

In a recent EEG study on program comprehension [26], participants also had to read code snippets derived from the study by Siegmund and others [40]. The results showed that BAs 6 and 44 were activated, but they did not report data on other activated areas.

All of this evidence allows us to answer RQ1:

We **can** largely replicate the results of the study by Siegmund and others.

RQ2: What is the difference between bottom-up program comprehension and comprehension with semantic cues in terms of activation and brain areas involved?

Data. Figure 2 shows the activated BAs, which contrast semantic-cues with bottom-up comprehension. The areas largely overlap with those defined in the original study [40]. Specifically, BAs 21 and 44 in the brain's left hemisphere are also activated. We also found a significant effect in BA 6, albeit at a slightly different location

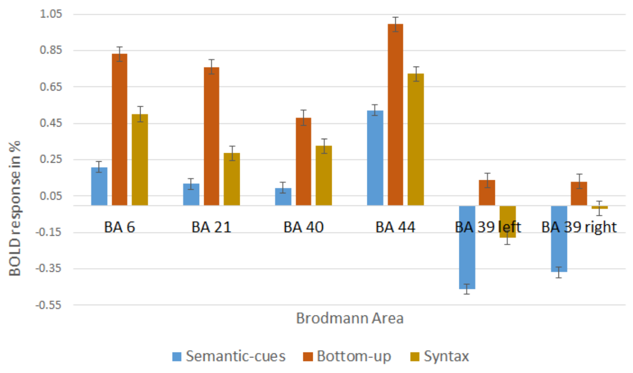

Figure 3: Average BOLD response. BA 39 is activated/deactivated in both hemispheres, BAs 6, 21, 40, and 44 in the left hemisphere. The whiskers indicate the standard error of the mean of the activation in the different participants.

compared to the previous study. The activation level during comprehension based on seamntic cues for all three areas is, however, significantly lower than for bottom-up comprehension.

Figure 3 shows the activation averaged across the code snippets for each condition (cf. Section 3). For example, the left group of bars indicates the activation of BA 6 for 3 individual conditions, that is, semantic-cues, bottom-up, and syntax errors. A positive value indicates an activation. A negative value indicates a deactivation in that area during the task, when compared to the average value across the whole session (which includes resting time).

The BOLD response for BA 39 in both hemispheres shows a *deactivation* during comprehension based on semantic cues and *activation* during bottom-up comprehension. Thus, BA 39 showed significantly less activation for semantic-cues than for bottom-up comprehension as well as during rest (cf. Section 4).

Even though the direct contrast between bottom-up and semantic-cues did not reveal significant activation in BA 40 at the conservative significance level, we show values for this region as defined in the original study, because we found a significant difference between bottom-up and syntax. This indicates that there is also a considerable difference between the bottom-up and the semantic-cues conditions.

Discussion. The activation of the BAs 6, 21, 40, and 44 indicates that, for comprehension based on semantic cues, all participants perform the same cognitive activities, that is, extracting the meaning of words and symbols and combining them to create a general understanding of a snippet's purpose. There is evidence of a similarity between semantic-cues and bottom-up comprehension at a basic level, which implies a similarity between both processes. However, we expected to observe activation in memory-related areas, since theory suggests that comprehension based on semantic cues should activate programming plans. Several possibilities exist: 1) Our participants may not have formed programming plans in their minds due to low levels of expertise, 2) programming plans are embedded in the same neural circuits as comprehension so we cannot see differences in our study, or 3) the theory is simply

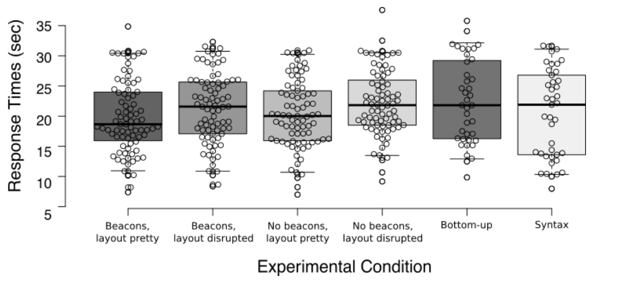

Figure 4: Response times in seconds per condition.

wrong. In future studies, we will dig deeper into the lack of memory retrieval during comprehension based on semantic cues.

The lower activation strength in these brain areas corresponds to better neural efficiency and indicates lower cognitive load. Thus, we assert that program comprehension based on semantic cues requires less cognitive effort than bottom-up comprehension. This difference is not caused by differences in task length, since all tasks lasted for 30 seconds. Looking at the response times of participants in Figure 4, we see no difference between the experimental conditions. The lower activation for comprehension based on semantic cues aligns well with our understanding of both comprehension processes suggesting that comprehension based on semantic cues is more efficient than bottom-up comprehension. Note that this is not reflected in the response times of participants, since we designed the semantic-cues snippets to require the same time as the bottom-up ones. Evidence from other neuro-imaging studies agrees with our result. Crk and Kluthe found that expertise leads to lower EEG signals (better neural efficiency), indicating lowered cognitive load [11]. In our study, we primed the participants with the code snippets (cf. Section 3), resulting in lower activation levels, and indicating lowered cognitive load.

In BA 39, comprehension based on semantic cues led to less activity compared with bottom-up comprehension and even the resting condition. This particular brain area has been shown to be part of the default mode network [34], which is active during resting periods between cognitively demanding tasks (e.g., [47]). Thus, the task context of program comprehension induces cognitive processes during the resting condition (i.e., between the program-comprehension tasks). The fact that the study by Siegmund and others did not mention this brain area can be explained by the roughly similar levels of activity in the bottom-up and the syntax condition. Underestimating which brain areas are involved in cognitive tasks is a common problem, especially for areas belonging to the default brain network [45]. Thus, future fMRI studies must be extra careful when devising their control conditions. Since BA 39 has been suggested to be involved in a number of cognitive processes, such as semantic processing, word reading and comprehension, number processing, memory retrieval, attention, and reasoning (which all seem relevant for all tasks of the current study), it is crucial to replace the resting condition with tasks that *distract* participants from activities related to program comprehension.

It is important to discern the degree of involvement of BA 39 in program comprehension, because it acts as a cross-modal hub, in which converging multisensory information is combined and integrated to comprehend and give sense to events, manipulate mental representations, solve familiar problems, and reorient attention to relevant information [38]. A promising starting point is to compare program comprehension with natural-language reading comprehension—the first function attributed to the angular gyrus [13, 21]. Regarding deactivation relative to rest, it has been suggested that reading affords fewer semantic associations than free associating [5]. This may also be true for program comprehension as applied to the current study.

Our control condition (locating syntax errors) was designed to subtract activation related to visual processing of the source code. We intended to avoid program comprehension by employing syntax errors (e.g., a missing closing bracket) that could be completed by a simple visual search. However, we did not find a difference in the activation pattern between comprehension based on semantic cues and syntax-error location.

On the one hand, it could mean that comprehension based on semantic cues is very similar to locating syntax errors. Looking at the tasks, it could be argued that deciding whether a snippet is familiar to participants requires pattern matching rather than full comprehension. On the other hand, the result could indicate that for locating syntax errors, participants at least partially comprehended the source code. Evidence from other studies shows that developers typically deploy an as-needed strategy. That is, they use understanding only as necessary to get a task done [36]. This validates the feedback we received from participants, who said they were not able to ignore the functionality of the code when reading it while locating errors. Syntax-error finding may mask other cognitive processes related to semantic cues. We will look into alternative control tasks, such as the bottom-up task, for future studies.

Thus, we can answer RQ2:

Comprehension based on semantic cues activates the same regions as bottom-up comprehension, except for BA 39, which is *deactivated* during semantic-cues comprehension, but activated during bottom-up comprehension. For all areas, the activation is significantly lower for comprehension based on semantic cues than for bottom-up comprehension.

RQ3: How do layout and beacons in source code influence program comprehension?

Data. Finally, we compare our 4 different semantic-cues comprehension conditions with one another. Overall, we found no significant differences, as supported by similar values of activation in the brain areas depicted in Figure 5.

Discussion. We could not find any influence of beacons and layout on program comprehension. This might indicate that the role of beacons and layout for program compehension is different than previous studies suggest [7, 28]. One reason may be due to our operationalization of semantic cues, which may have encouraged simple recognition of familiar snippets rather than comprehension. The demand of program comprehension processes may have been too low due to the intense engagement with highly comparable code snippets immediately before the fMRI experiment. Nevertheless, the results make clear that the effects resulting from different

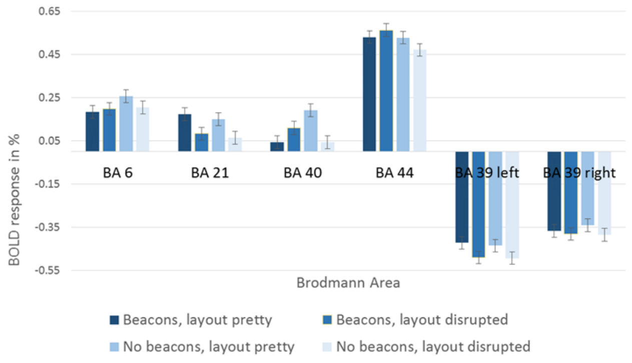

Figure 5: Average BOLD response. BA 39 is deactivated in both hemispheres, BAs 6, 21, 40, and 44 are activated in the left hemisphere. The whiskers indicate the standard error of the mean of the activation in the different participants.

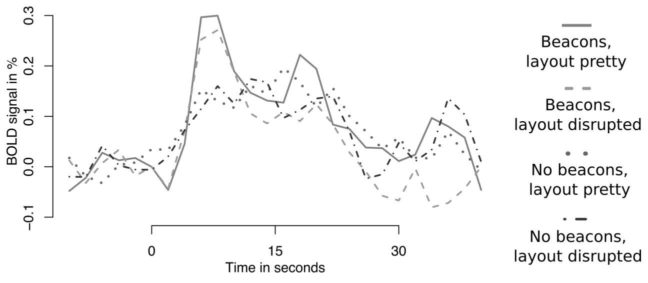

Figure 6: BOLD response per condition for BA 21.

efforts to foster comprehension seem to be small. We could increase the experimental sensitivity by studying a larger number of participants.

Still, we observed activation differences at specific time points of comprehending the code snippets. For example, looking at the BOLD response for BA 21 in Figure 6 for each of the 4 semantic-cues conditions, we see that the versions with the beacons both have a peak a few seconds after task onset, independent of whether the layout was pretty-printed or disrupted. The temporal differences may indicate differences in how long a programmer may spend in specific phases of a cognitive process. It would be interesting to explore whether one of these effect manifests in further studies, because BA 21 is a classical natural-language processing area. More introspective information about the dynamics during the program comprehension are needed. Eye tracking during fMRI acquisition may provide additional information and should be integrated in future studies.

Neither beacons nor program layout seem to significantly affect the program comprehension process.

5 PERSPECTIVES

5.1 Relation to Theories of Comprehension

The different activation patterns we see between semantic-cues and bottom-up comprehension provide evidence in support of some aspects of theories of program comprehension, while casting doubt on others.

Semantic Chunking. The theory of bottom-up comprehension suggests that developers start with details of source code and group these to semantic chunks, until they gain a high-level understanding of the program [39]. Based on the activation of BA 39 during bottom-up comprehension, and its role as an integration hub for semantic information, the evidence supports semantic chunking. Its absence during comprehension based on semantic cues is consistent with successive hypothesis refinement [8].

Neural Efficiency. Studies of brain activity find that experts demonstrate more efficient neuronal firing patterns than novices with the same tasks [30]. More experienced programmers also demonstrate lower activation than less experienced programmers [11, 12]. Similarly, we have observed reduced activation during comprehension based on semantic cues. Our evidence is consistent with the view that semantic cues reduce cognitive load.

Plans. Software-engineering researchers have proposed that programmers use knowledge structures [35] called *plans* that encode semantic and domain information about a program [8]. Researchers theorized that programmers activate plans containing a high-level schema of typical program structures when they find evidence in support of their hypothesis about the code's execution. Evidence often comes in the form of beacons, such as method names and stereotypical code sequences. Finally, it was proposed that programs follow basic rules of discourse, and that any violation to "accepted conventions of programming" would hamper an experienced programmer's ability to use plans [44].

We found evidence that is inconsistent with some theoretical aspects of plans. Unlike previous experiments that manipulated syntax layout and beacons [44], we could not confirm any influence on the cognitive processes developers used when comprehending source code. Even if the code is scrambled or the layout is changed, a programmer's cognitive processes may be robust enough to deal with them, just as programmers can understand code that is incomplete or has syntax errors. We did not find any specific cognitive processes that would be consistent with plan activation outside of the program comprehension process. An alternative theory may be that, if plans exist, they are embedded in the same circuits used during program comprehension. Overall reduced activation in the network of program-comprehension brain areas may then be the result of plan activation.

5.2 Implications and Future Directions

Tool Support. A recent fMRI study by Sato and others found that having access to Euler diagrams during logical-reasoning tasks allowed participants to offload the content of their working memory used to represent the logical statements onto the diagram itself [37]. As a result, this freed up resources to solve the reasoning tasks faster. The ability to support *cognitive offloading* has several important consequences. For example, the right hemisphere of the brain can take on a secondary task if both the primary task and secondary task are simple and do not require access to the same types of information [10]. Both hemispheres are recruited in complex tasks [3]. The diagram used by Sato and others would enable a person to perform more complex logical-reasoning tasks, because they can offload the representation onto the diagram. Likewise, if a programming tool is able to free up cognitive resources, such as a visualization or debugger tool, then programmers may be able to perform increasingly complex cognitive tasks, both that were not possible before, and with a reduced chance of mental errors. As we found in our work, the programmer is better able to integrate information when they do not have to perform semantic chunking. If a tool could be found to reduce the need to activate a particular brain region, then direct evidence could be offered about the ability of a tool to reduce cognitive load and potential mental errors. In the long run, this will help us and other researchers to develop tools (e.g., those similar to debuggers that show how values of variables change, but which are more customizable) that support developers in relieving cognitive resources. Developers would then be able to focus more on the actual task at hand, without being restricted by their own cognitive limits.

Experimental Paradigm. Besides successfully replicating the results of the study by Siegmund and others, we went beyond the state of the art by providing new evidence regarding the neural efficiency of comprehension based on semantic cues. This paves the way for fMRI and other neuro-imaging techniques to be used in future research on program comprehension and related cognitive processes. We wish to show that the lengthy process of conducting series of pilot studies is worth the effort, so we can add a neuro-scientific perspective to the understanding of human factors in software engineering. With neuro-imaging studies becoming more and more prevalent in software-engineering research (cf. Section 7), our study makes an important contribution toward establishing this modality as standard measurement instrument.

Future Studies. As a further avenue of research, we would like to explore which aspects of source code have significant influence on the comprehension process. In our study, we began this endeavor by looking at beacons and layout, which prior work focused on. In future studies, researchers should look at also other aspects of code, such as patterns [27] and plans [35], and at various operationalizations of top-down program comprehension. This would help the community to gain a more holistic understanding about program comprehension. In the long run, this will allow us and other researchers to derive new rules for how to semantically and syntactically structure source code (for example, to highlight beacons in source code). Furthermore, we hope that we can improve programming education by helping novices to focus on relevant parts of source code (e.g., to more quickly learn to identify and make use of beacons).

6 THREATS TO VALIDITY

6.1 Internal Validity

The operationalization of comprehension based on semantic cues is closely linked with recognition and may not require much comprehension. This might explain why we did not observe a difference in activation between comprehension based on semantic cues and syntax-error finding. However, since recognition is also important for comprehension based on semantic cues, and because our participants indicated that they did read the source code to locate syntax errors, we argue that our current operationalization of comprehension based on semantic cues was suitable for the study.

Identifying BAs based on Talairach coordinates (used by scanner software to identify voxels) requires a lot of expertise. Other researchers may attribute the same activated clusters of voxels to nearby Brodmann areas. However, our team has considerable experience with mapping voxels to Brodmann areas, and we double-checked the assignments. Thus, we believe we have this threat under control.

6.2 External Validity

Our limited experiment setup cannot generalize to program comprehension in different settings (for example, large software projects). Additionally, our operationalization of comprehension based on semantic cues captures only one aspect of this complex process, meaning we may have missed some. This is a fundamental trade-off in empirical studies, where we have to control external influences as much as possible (internal validity), or strive for a more generalizable experimental setting (external validity) [42]. Since we looked at both bottom-up and semantic-cues comprehension, we increased the external validity of the study by Siegmund and others.

7 RELATED WORK

In recent years, several neuro-imaging techniques have been employed in software-engineering research. The first fMRI study in this context was conducted by Siegmund and others [40], as we have discussed.

Directly inspired by that study, Floyd and others also conducted an fMRI study, but focused on a comparison of the representation of programming and natural languages [17]. In their study, they had developers review code and prose in an fMRI scanner, and then compared brain activation. They used the activation patterns to successfully predict which tasks participants were completing. This is different from our study goal, which is to understand which brain areas are activated during semantic-cues program comprehension.

Another fMRI study conducted by Duraes and others focused on locating defects in software [15]. The authors found a stronger activation in the right anterior insula, depending on whether a bug was initially spotted compared to when it is confirmed.

Nakagawa and others used another neuro-imaging technique, called functional near-infrared spectroscopy (fNIRS), to evaluate the cerebral blood flow during mental code execution [29]. They found increased blood flow in the prefrontal cortex that correlated with the difficulty of a task. Ikutani and Uwano also used fNIRS in a comprehension study and found an increased activation in the frontal pole when participants memorized variable names without manipulating their values [24].

Crk and Kluthe used electroencephalography (EEG) to observe the influence of programmer expertise on EEG activation patterns during program-comprehension tasks [11]. They found that lower expertise led to higher cognitive load. Lee and others also used EEG in a setting similar to the fMRI study by Siegmund and others [26]. Their study found a subset of the same brain areas as we did, which also Siegmund and others found, that is, BAs 6 and 44. This confirms the role of these areas in bottom-up program comprehension.

Parnin used electromyography (EMG) to measure subvocalization (i.e., inner monologues) during programming [31]. He found that developers use different levels of subvocalization for each task, such as debugging and testing.

Many of these studies are explorative in nature. They evaluate how particular neuro-imaging techniques can be applied in the software-engineering domain. They all serve as a good starting point to establish neuro-imaging techniques to better understand human factors in software engineering.

In addition to single techniques, researchers have tried combining several measures. For example, Fritz and others [20] used a setup in which they combined several psycho-physiological measures: EEG, eye tracking, and electrodermal activity. Participants performed mental execution of code while instrumented with sensors. The authors successfully used the combination of all three sensor readings to predict the perceived task difficulty of the participants. Lee and others used a combination of EEG and eye tracking to predict task difficulty and programmer expertise [25]. They found that both sensors, alone and in combination, could predict expertise and task difficulty. Much like the study by Floyd and others [17], the goal of these multi-sensor studies is to use the sensor data to predict task difficulty or cognitive load.

8 CONCLUSION

Program comprehension based on semantic cues is a very efficient process for understanding source code compared with the tedious, statement-by-statement process employed during bottom-up comprehension. In this paper, we replicated an fMRI study to deepen our understanding of program comprehension. First, we were able to replicate the results of the previous fMRI study, confirming the role of several Brodmann areas and related cognitive processes for bottom-up program comprehension. This strengthens the role of fMRI as important measurement instrument in software engineering research. Second, we found that program comprehension based on semantic cues leads to a lower activation intensity as compared to bottom-up comprehension, increasing support from a neuroscience perspective for the hypothesis that comprehension based on semantic cues leads to neural efficiency. Finally, we found no evidence that beacons or program layout affect the comprehension process. However, it may be that the effect is just too small to detect with our experimental setup.

ACKNOWLEDGMENTS

We thank Anke Michalsky, Andreas Fügner, and Jörg Stadler for fMRI data acquisition and all participants of our pilot studies and the fMRI study. Kästner's work has been supported by NSF awards 1318808 and 1552944, and Apel's work by the DFG grants AP 206/4 and AP 206/6. Siegmund's and Brechmann's work is supported by DFG grant SI 2045/2-1. Siegmund's work is further funded by the Bavarian State Ministry of Education, Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B).

REFERENCES

[1] And Turken and Nina Dronkers. 2011. The Neural Architecture of the Language Comprehension Network: Converging Evidence from Lesion and Connectivity Analyses. *Frontiers in Systems Neuroscience* 5, 1 (2011).

[2] Jörg Bahlmann, Ricarda Schubotz, and Angela Friederici. 2008. Hierarchical Artificial Grammar Processing Engages Broca's Area. *NeuroImage* 42, 2 (2008), 525–534.

[3] Marie T Banich. 1998. The missing link: the role of interhemispheric interaction in attentional processing. *Brain and Cognition* 36, 2 (March 1998), 128–157.

[4] Jeffrey Binder, Rutvik Desai, William Graves, and Lisa Conant. 2009. Where Is the Semantic System? A Critical Review and Meta-Analysis of 120 Functional Neuroimaging Studies. *Cerebral Cortex* 19, 12 (2009), 2767–2796.

[5] Jeffrey R Binder, Julia A Frost, Thomas A Hammeke, PSF Bellgowan, Stephen M Rao, and Robert W Cox. 1999. Conceptual Processing during the Conscious Resting State: A Functional MRI Study. *J. Cognitive Neuroscience* 11, 1 (1999), 80–95.

[6] Korbinian Brodmann. 2006. *Brodmann's Localisation in the Cerebral Cortex.* Springer.

[7] Ruven Brooks. 1978. Using a Behavioral Theory of Program Comprehension in Software Engineering. In *Proc. Int. Conf. Software Engineering (ICSE).* IEEE, 196–201.

[8] Ruven Brooks. 1983. Towards a Theory of the Comprehension of Computer Programs. *Int. J. Man-Machine Studies* 18, 6 (1983), 543–554.

[9] B Chance, Z Zhuang, C UnAh, C Alter, and Lipton L. 1993. Cognition-Activated Low-Frequency Modulation of Light Absorption in Human Brain. *Proc. Nat. Academy Sciences of the United States of America (PNAS)* 90, 8 (1993), 3770–3774.

[10] Sylvain Charron and Etienne Koechlin. 2010. Divided Representation of Concurrent Goals in the Human Frontal Lobes. *Science* 328, 5976 (16 April 2010), 360–363.

[11] Igor Crk and Timothy Kluthe. 2014. Toward Using Alpha and Theta Brain Waves to Quantify Programmer Expertise. In *Proc. Int. Conf. Engineering in Medicine and Biology Society (EMBC).* IEEE, 5373–5376.

[12] Igor Crk, Timothy Kluthe, and Andreas Stefik. 2015. Understanding Programming Expertise: An Empirical Study of Phasic Brain Wave Changes. *ACM Trans. Comput.-Hum. Interact.* 23, 1, Article 2 (Dec. 2015), 29 pages.

[13] J Dejerine. 1891. Sur un cas de Cecite Verbale avec Agraphie, Suivi d'Autopsie. *C. R. Societe de Biologie* 43 (1891), 197–201.

[14] Nina F Dronkers, David P Wilkins, Robert D Van Valin, Brenda B Redfern, and Jeri J Jaeger. 2004. Lesion Analysis of the Brain Areas Involved in Language Comprehension. *Cognition* 92, 1–2 (2004), 145–177.

[15] João Duraes, Henrique Madeira, J Castelhano, C Duarte, and M Castelo Branco. 2016. WAP: Understanding the Brain at Software Debugging. In *Proc. Int. Symp. Software Reliability Engineering (ISSRE).* IEEE, 87–92.

[16] Christian J Fiebach, Matthias Schlesewsky, Gabriele Lohmann, D Yves Von Cramon, and Angela D Friederici. 2005. Revisiting the Role of Broca's Area in Sentence Processing: Syntactic Integration Versus Syntactic Working Memory. *Human Brain Mapping* 24, 2 (2005), 79–91.

[17] Benjamin Floyd, Tyler Santander, and Westley Weimer. 2017. Decoding the Representation of Code in the Brain: An fMRI Study of Code Review and Expertise. In *Proc. Int. Conf. Software Engineering (ICSE).* IEEE.

[18] Angela D Friederici. 2002. Towards a Neural Basis of Auditory Sentence Processing. *Trends in Cognitive Sciences* 6, 2 (2002), 78–84.

[19] Angela D Friederici and Sonja A Kotz. 2003. The Brain Basis of Syntactic Processes: Functional Imaging and Lesion Studies. *NeuroImage* 20, 1 (2003), S8–S17.

[20] Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using Psycho-physiological Measures to Assess Task Difficulty in Software Development. In *Proc. Int. Conf. Software Engineering (ICSE).* ACM, 402–413.

[21] Norman Geschwind. 1965. Disconnexion Syndromes in Animals and Man. *Brain* 88, 2 (1965), 237–294.

[22] Yosef Grodzinsky and Andrea Santi. 2008. The Battle for Broca's Region. *Trends in Cognitive Sciences* 12, 12 (2008), 474–480.

[23] Scott A Huettel, Allen W Song, and Gregory McCarthy. 2004. *Functional Magnetic Resonance Imaging.* Vol. 1. Sinauer Associates Sunderland.

[24] Yoshiharu Ikutani and Hidetake Uwano. 2014. Brain Activity Measurement during Program Comprehension with NIRS. In *Proc. Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD).* IEEE, 1–6.

[25] Seolhwa Lee, Danial Hooshyar, Hyesung Ji, Kichun Nam, and Heuiseok Lim. 2017. Mining Biometric Data to Predict Programmer Expertise and Task Difficulty. *Cluster Computing* (2017), 1–11.

[26] SeolHwa Lee, Andrew Matteson, Danial Hooshyar, SongHyun Kim, JaeBum Jung, GiChun Nam, and HeuiSeok Lim. 2016. Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis. In *Proc. Int. Conf. Bioinformatics and Bioengineering (BIBE).* IEEE, 350–355.

[27] Richard C Linger, Harlan D Mills, and Bernard I Witt. 1979. Structured Programming: Theory and Practice. (1979).

[28] Richard J. Miara, Joyce A. Musselman, Juan A. Navarro, and Ben Shneiderman. 1983. Program Indentation and Comprehensibility. *Commun. ACM* 26, 11 (1983), 861–867.

[29] Takao Nakagawa, Yasutaka Kamei, Hidetake Uwano, Akito Monden, Kenichi Matsumoto, and Daniel M. German. 2014. Quantifying Programmers' Mental Workload During Program Comprehension Based on Cerebral Blood Flow Measurement: A Controlled Experiment. In *Proc. Int. Conf. Software Engineering (ICSE).* ACM, 448–451.

[30] Aljoscha C. Neubauer and Andreas Fink. 2009. Intelligence and neural efficiency. *Neuroscience and Biobehavioral Reviews* 33, 7 (10 July 2009), 1004–1023.

[31] Chris Parnin. 2011. Subvocalization — Toward Hearing the Inner Thoughts of Developers. In *Proc. Int. Conf. Program Comprehension (ICPC).* IEEE, 197–200.

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

[33] Karl Petersson, Vasiliki Folia, and Peter Hagoort. 2012. What Artificial Grammar Learning Reveals about the Neurobiology of Syntax. *Brain and Language* 298, 1089 (2012), 199–209.

[34] Marcus Raichle, Ann MacLeod, Abraham Snyder, William Powers, Debra Gusnard, and Gordon Shulman. 2001. A Default Mode of Brain Function. *Proc. Nat. Academy of Sciences* 98, 2 (2001), 676–682.

[35] Charles Rich. 1981. *Inspection Methods in Programming.* Technical Report TR-604. MIT.

[36] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How Do Professional Developers Comprehend Software?. In *Proc. Int. Conf. Software Engineering (ICSE).* IEEE, 255–265.

[37] Yuri Sato, Sayako Masuda, Yoshiaki Someya, Takeo Tsujii, and Shigeru Watanabe. 2015. An fMRI Analysis of the Efficacy of Euler Diagrams in Logical Reasoning. In *Proc. Symp. on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, 143–151.

[38] Mohamed L Seghier. 2013. The Angular Gyrus: Multiple Functions and Multiple Subdivisions. *The Neuroscientist* 19, 1 (2013), 43–61.

[39] Ben Shneiderman and Richard Mayer. 1979. Syntactic/Semantic Interactions in Programmer Behavior: A Model and Experimental Results. *Int. J. Parallel Programming* 8, 3 (1979), 219–238.

[40] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding Understanding Source Code with Functional Magnetic Resonance Imaging. In *Proc. Int. Conf. Software Engineering (ICSE).* IEEE, 378–389.

[41] Janet Siegmund, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2014. Measuring and Modeling Programming Experience. *Empirical Softw. Eng.* 19, 5 (Oct. 2014), 1299–1334.

[42] Janet Siegmund, Norbert Siegmund, and Sven Apel. 2015. Views on Internal and External Validity in Empirical Software Engineering. In *Proc. Int. Conf. Software Engineering (ICSE),* Vol. 1. IEEE, 9–19.

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

[44] Elliot Soloway and Kate Ehrlich. 1984. Empirical Studies of Programming Knowledge. *IEEE Trans. Softw. Eng.* 10, 5 (1984), 595–609.

[45] Craig EL Stark and Larry R Squire. 2001. When Zero is Not Zero: The Problem of Ambiguous Baseline Conditions in fMRI. *Proc. Nat. Academy Sciences of the United States of America (PNAS)* 98, 22 (2001), 12760–12766.

[46] Jean Talairach and Pierre Tournoux. 1988. *Co-Planar Stereotaxic Atlas of the Human Brain.* Thieme.

[47] Miranka Wirth, Kay Jann, Thomas Dierks, Andrea Federspiel, Roland Wiest, and Helge Horn. 2011. Semantic Memory Involvement in the Default Mode Network: A Functional Neuroimaging Study Using Independent Component Analysis. *NeuroImage* 54, 4 (2011), 3057–3066.