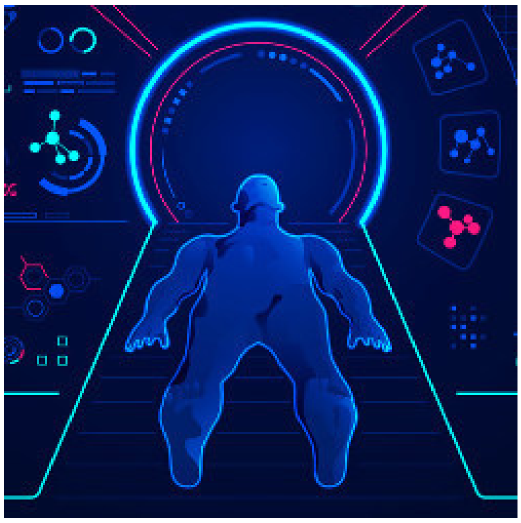
Studying Programming in the Neuroage: Just a Crazy Idea?

cacm.acm.org/magazines/2020/6/245156-studying-programming-in-the-neuroage/fulltext

By Janet Siegmund, Norman Peitek, André Brechmann, Chris Parnin, Sven Apel
Communications of the ACM, June 2020, Vol. 63 No. 6, Pages 30-34
10.1145/3347093

This is a crazy idea," the review read. Closing my laptop lid, I added in my mind "and ... it will never work," as a lump welled in my throat. What we were proposing to do was simple yet ambitious. Using functional magnetic resonance imaging, we might better understand what goes on in the minds of programmers as they read and understand code. We had performed pilot experiments with a neurobiologist, had promising results, and encouraging words from colleagues and reviewers. Still, the words, "this is a crazy idea," echoed in our minds. Would it be possible to break open the stale progress in program comprehension research? After all, researchers have been working on understanding programmers since the 1970s. Could neuroimaging really help devising a conclusive and comprehensive theory of program comprehension?


Credit: Jackie Niam

The Waves of Program Comprehension Research

Research in program comprehension has been a cycle of booms and busts. In the early 1970s and 1980s, the first wave of researchers were psychologists, using methods, such as memory recall, to probe how programmers represent and process code in their mind. As a result, various theories and mechanisms were proposed, such as programming plans and bottom-up comprehension, but no clear consensus and only few impulses for programming research (including research on programming methodology, language design, or education) emerged.

More than a decade passed without significant research progress and many leaving the field.[25] In the mid-2000s, a second wave of researchers emerged, now with "big code" as the methodology of choice. Researchers mined traces of program comprehension as manifested through programming activities in code repositories, such as GitHub, asking statistical questions, such as whether long or short identifier names lead to more defects. While this data has proved valuable, the community was drifting farther and farther away from understanding the inner workings of the programmer's mind, and so did our ability to devise and validate a conclusive and explanatory theory of program comprehension.

Meanwhile, in the field of psychology and cognitive neuroscience, considerable progress has been made in building theories of fundamental cognitive processes, such as language comprehension and logical reasoning. One important enabling technology was brain imaging, including functional magnetic resonance imaging (fMRI), electroencephalography (EEG), and functional near infrared spectroscopy (fNIRS). These methods have revolutionized the understanding of cognitive processes and are routinely used as a measurement tool in various disciplines, including psychology, economics, and social sciences.[24]

From Idea to Implementation

Excited by the developments in brain imaging and its successful application in other disciplines, the idea grew to revitalize the research efforts of program comprehension. The first step was to build a team. We had to make several pitches to different scientists, before we finally found a neurobiologist who would listen to our idea. With him (the third author) on board, we started by absorbing the relevant neuroscience literature, which required several years of investigation and an in-depth understanding of brain imaging methods and what they can or cannot do.[24] Most notably, brain-imaging methods cannot directly observe cognitive processes, but only their neurobiological correlates, which poses limitations on measurement resolution, experiment design, and interpretability of results. For example, fMRI measures the change in the oxygenation level of blood as brain activity occurs based on different magnetic properties of oxygenated and deoxygenated blood. Oxygenation changes take time, adding a lag of a few seconds (that is, the hemodynamic lag[12]). This limits the temporal resolution of fMRI to the order of seconds. As further constraint, one needs a baseline level of activation as comparison to detect changes in the first place. Also financial constraints play a role (one hour in our fMRI scanner costs about 150 €).

Further challenges lie in the experimental design. First, one needs to develop a set of tasks that could be performed by programmers to measure the act (and only the act) of program comprehension. This is in itself challenging, as there is no canonical set of programming problems. Additionally, the tasks need to respect the limitations of brain imaging methods, which affect code length and the time to accomplish the task. For example, only 20 lines of code can be reasonably shown without allowing scrolling, and they require about 30 to 60 seconds to understand (depending on complexity), so we used 60 seconds as time limit for each snippet. Longer source code that requires scrolling and longer task duration is also possible in today's scanners. After many discussions and several pilot studies, we arrived at a set of general program comprehension tasks covering basic (imperative) program structures and programming problems inspired by introductory programming courses and textbooks.

A second key ingredient is a method to subtract brain activity related to program comprehension from brain activity unrelated to program comprehension, for example, related to reading, speaking, or motor activity. If we just observe programmers while they work with source code, we see a lot of activated brain areas, but we do not know which are directly related to the act of program comprehension. To determine which part of the brain is specifically activated during program comprehension, we apply a subtraction method,[5] illustrated in Figure 2: We let the programmer identify syntax errors in code as a baseline task, called control condition, which reveals the difference between "glancing through" the code as compared to deeply understanding its semantics. This subtraction method is conservative in order not to discover spurious activations, such that, activation not related to the act of program comprehension is filtered out as much as possible.

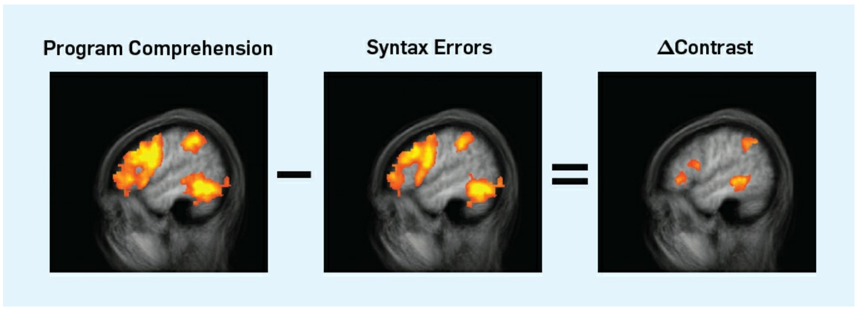
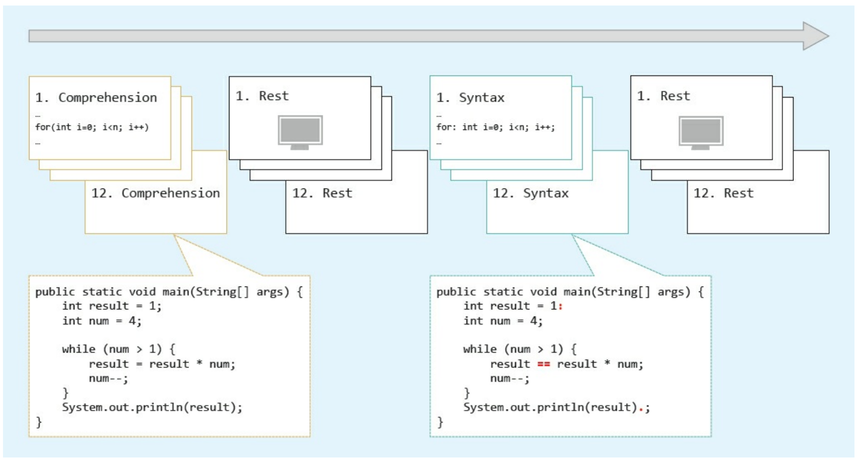


Figure 2. The subtraction method: Program comprehension triggers also unrelated brain activation, so we subtract brain activation during identifying syntax errors to obtain brain activation specific to program comprehension.

After several years of planning and many pilot tests outside an fMRI scanner, we had arrived at an experimental design that could be executed inside the fMRI machine. We presented our plans at the New Ideas and Emerging Results track at ESEC/FSE in 2012, receiving very diverse feedback, ranging from "oh wow, this is so cool" to "that will never work."

The First Program Comprehension Study of Its Kind

We conducted our first study with 17 students in an fMRI scanner understanding a dozen code snippets (Figure 1 provides an overview of the experiment procedure), all well prepared in terms of difficulty and duration needed to understand by the average programmer.[26] The results indicated that a specific network of brain areas in the left hemisphere was used by participants to understand code, including areas related to working memory, divided attention, and reading comprehension. Surprisingly, we did not observe cognitive processes related to mathematical and logical reasoning, which would be consistent with the perspective that programming is a formal, logical, and mathematical process. The most striking result is a clear left-lateral activation during program comprehension. In conjunction with the activation in semantically associated areas (BAs 21 and 47), this suggests that program comprehension involves semantic processes that are also relevant in language processing. Although neuroscience researchers still argue about the differential role of the two hemispheres in semantic processing in general, there is clear evidence on the dominant role of the left hemisphere in language-related processing. If the role of the left hemisphere in program comprehension can be consolidated in future studies, this would strengthen Dijkstra's conjecture that a proper education in natural languages is imperative for successful programmers.[4]



```
…
for(int i=0; i<n; i++)
…
```

```
…
for: int i=0; i<n; i++;
…
```

```
public static void main(String[] args) {
    int result = 1;
    int num = 4;

    while (num > 1) {
        result = result * num;
        num--;
    }
    System.out.println(result);
}
```

```
public static void main(String[] args) {
    int result = 1:
    int num = 4;

    while (num > 1) {
        result == result * num;
        num--;
    }
    System.out.println(result).;
}
```

Figure 1. Experiment design of the first fMRI study[26]: participants understood a source code snippet (60s), followed by a rest period to let the oxygenation level return to the baseline (30s), followed by identifying syntax errors (30s), and another rest period (30s); this process was repeated for 12 times.

Further Studies

Although standard in neuroscience, the complexity of the process and machinery led to the question of how reliable the results are—doubts were rising. Following best practice in neuroscience, we replicated our study twice with varying participants, comprehension tasks, and source code snippets. Especially, the small initial sample drove us to collect more data to increase statistical power. However, as more studies are following with more specific questions, leading to smaller differences in conditions, we need to increase the sample size. Unfortunately, without upfront experience about the expected effect size, it is difficult to recommend a sample size. In the replications, we reliably found activation in the same brain areas,[23,27] which increased our confidence in the validity of our setup and results. This reliability of the setup is possibly even more valuable than the individual results we obtained so far.

> *Could neuroimaging really help devising a conclusive and comprehensive theory of program comprehension?*

Further evidence on the validity of our results came also from other research teams, who replicated and extended our experimental design. Inspired by our study, Floyd and others changed the contrast condition to review source code (instead of finding syntax errors) and added a third condition to review natural-language prose.[8] They could also replicate a subset of the brain areas that we found. Lee and others replicated our results with EEG. In contrast to fMRI, EEG directly measures the electrical responses of neurons with high temporal resolution (that is, in the order of milliseconds), but with a limited spatial resolution. They found brain areas that we also found, with the additional insight that programming expertise modulates the respective activation strength.[17]

Despite all these results, it is important to note that typically, no brain area is specifically associated with one cognitive process, but with several. For example, BAs 21, 44, and 47 are also associated with spoken language, but since we did not require participants to give a spoken response, it is not relevant for our setting. Open databases, such as BrainMap or Neurosynth, let us identify which tasks and cognitive processes are associated with a selected brain region (Genon and others give a summary of how we can map cognitive processes to brain areas[10]).

The Third Wave of Program Comprehension Research

Other research teams also adopted a neurocognitive perspective on program comprehension: Kosti and others used our experiment design to record EEG data and found a correlation of EEG activation with subjective rating of difficulty.[15] Studies using fNIRS found activation in frontal parts of the brain depending on the kind of task (memorizing variable names vs. mental arithmetic)[14] and task difficulty.[20] Duraes and others used fMRI to observe the neural activation during the location of code defects, identifying a specific activity pattern when a bug was spotted and confirmed.[6] Studies using EEG revealed a relationship between mental load and expertise[3,28] such that with lower expertise, the same tasks require higher mental load. This is because experts have developed an efficient cognitive representation, so they can manage more information without requiring a larger working memory capacity. This can be observed in many areas other than program comprehension, for example, with expert chess players or superior memorizers. They do not have an exceptionally large working memory capacity, yet they can recall large amount of information by using specific encoding strategies, that is, certain configurations that imply the positions of chess figures[2] or visual objects linked to spatial landmarks.[18] Compared to non-experts, this is reflected in a less pronounced or different activation pattern that develops concomitant with their growing expertise.[19] The mental load is also reflected in the response of the default mode network, which is typically activated during self-referential processing and deactivated during a cognitively demanding task to avoid interference. The higher the cognitive load, the stronger the deactivation.[9] We observed this effect also in our recent study, such that more complex tasks are related with a stronger deactivation of the default mode network.[22]

A Multimodal Approach

Despite promising results, the community has realized quickly that a single imaging or measurement method is not sufficient to understand the full complexity of the cognitive processes involved in program comprehension and, more importantly, to connect the cognitive processes to the programmer's behavior. Only a multimodal approach is able to achieve this. As an example, a recent study[23] suggests that the integration of eye tracking and fMRI is able to draw this connection by connecting eye movement patterns and brain activation to the programmer's strategy of program comprehension (the eye movement pattern observed suggests that programmers follow program identifiers during top-down comprehension). Fakhoury and others combined fNIRS and eye tracking showing that, when participants are examining unsuitable identifier names, cognitive load increases.[7] A multimodal setting is able to better identify expert programmers, as Lee and others showed by extending their EEG study with eye tracking.[16] Huang and others have investigated programmers' brains while manipulating data structures, such as binary trees, in contrast to mental rotations with fMRI and fNIRS. The study not only provided insights into programmers' cognitive processes, but also showed that fMRI is more sensitive for identifying cognitive processes than fNIRS (in terms of statistical power and spatial resolution).[11] Nevertheless, fNIRS may be an interesting alternative in certain settings, as it is cheaper, less restrictive, and easier to combine with other modalities.

> *Neuroimaging offers a unique opportunity to understand, build, and test theories of program comprehension like never before.*

fMRI and EEG can also be combined to complement their strengths and weaknesses.[13] For example, Bledowski and others mapped the stages of retrieving information from working memory to the fast neuronal response of different brain areas, shedding light on how the retrieval process unfolds temporally and spatially.[1]

fMRI not only lets us dive deeper into cognitive processes, but it also allows us to decode brain activity to predict the tasks (referred to as reverse inference).[24] Floyd and others trained a classifier based on the data of their replication study to predict the kind of task participants were completing (79% accuracy). Future studies might reveal whether specific aspects of a task (for example, words vs. numbers being manipulated in source code) are represented in specific brain areas or in different activation strengths.

Entering the Neuroage

Programming research has entered the Neuroage. Neuroimaging offers a unique opportunity to understand, build, and test theories of program comprehension like never before. It empowers researchers to obtain insights into the cognitive processes involved and their connection to the programmer's behavior. This way, the research community will be able to understand why, how, and to what extent program comprehension takes place, not only whether. For example, rather than stating that programmers tend to process loops faster than recursive structures, we would like to quantify how and to what extent the use of loops or recursion influences task completion time and which cognitive processes are responsible for this difference (for example, keeping track of intermediate results in recursive computations easily exceeds the programmer's working memory), and how this is modulated by expertise.

Although research has started off just a few years ago, the small but growing community already provided interesting insights into the cognitive processes of program comprehension. By adopting neuroimaging methods, researchers may overcome the stale progress of program comprehension by better understanding accompanying cognitive subprocesses. This progress can pave the way toward a theory of program comprehension. Maybe program comprehension is not a unique cognitive process, but just a special form of problem solving? Maybe cognitive modeling (for example, using cognitive architectures like ACT-R, which let researchers describe a cognitive process as a series of discrete operations[21]) can help to describe the fundamental cognitive processes underlying program comprehension? We cannot know, yet, but having demonstrated the validity and usefulness of neuroimaging methods for programming research, the real fun (and hard work) begins. With these methods, we might be able to understand why programmers make mistakes, why some code is more difficult than other, what programmer expertise looks like, and how we can best train and prepare programmers. Ultimately, these are steps forward to reach a more conclusive and comprehensive theory of program comprehension.

> *Maybe program comprehension is not a unique cognitive process, but just a special form of problem solving?*

We believe this research direction can also feed back to neuroscience research, such that we may discover new nuances of cognitive processes. Specifically, our setup requires participants to mentally execute code, so program comprehension is more than reading comprehension, and it might be different from other mental simulations, say thinking through a cooking recipe. We have reached an exciting new point in studying the human programmer, and we cannot wait to see where the research community will take it.

References

1. Bledowski, C. et al. Mental chronometry of working memory retrieval: A combined functional magnetic resonance imaging and event-related potentials approach. *Journal of Neuroscience 26*, 3 (2006), 821–829.

2. Chase, W. and Simon, H. Perception in Chess. *Cognitive Psychology 4*, 1 (1973), 55–81.

3. Crk, I., Kluthe, T., and Stefik, A. Understanding programming expertise: An empirical study of phasic brain wave changes. *ACM Transactions on Computer-Human Interaction 23*, 1 (2015), 1–29.

4. Dijkstra, E. *Selected Writings on Computing: A Personal Perspective*. Springer, 1982.

5. Donders, F. On the speed of mental processes. *Acta Psychologica 30*, 1 (1969), 412–431.

6. Duraes, J. et al. WAP: Understanding the brain at software debugging. In *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2016, 87–92.

7. Fakhoury, S. et al. The effect of poor source code lexicon and readability on developers' cognitive load. In *Proceedings of the International Conference on Program Comprehension (ICPC)*, ACM, 2018, 286–296.

8. Floyd, B., Santander, T., and Weimer, W. Decoding the representation of code in the brain: An fMRI study of code review and expertise. In *Proceedings of the International Conference on Software Engineering (ICSE)*, pages 175–186. IEEE, 2017.

9. Gazzaniga, M., Ivry, R., and Mangun, G. *Cognitive Neuroscience: The Biology of the Mind*. Norton & Company, 2013.

10. Genon, S. et al. How to characterize the function of a brain region. *Trends in Cognitive Sciences 22*, 4 (2018), 350–364.

11. Huang, Y. et al. Distilling neural representations of data structure manipulation using fMRI and fNIRS. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE, 2019.

12. Huettel, S., Song, A., and McCarthy, G. *Functional Magnetic Resonance Imaging, volume 3*. Sinauer Associates, 2014.

13. Huster, R. et al. Methods for simultaneous EEG-fMRI: An introductory review. *Journal of Neuroscience 32* ,18 (2012), 6053–6060.

14. Ikutani, Y. and Uwano, H. Brain Activity Measurement during program comprehension with NIRS. In *Proceedings of the International Conference Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, IEEE, 2014, 1–6.

15. Kosti, K. et al. Towards an affordable brain computer interface for the assessment of programmers' mental workload. J. *Human-Computer Studies 115*, (2018), 52–66.

16. Lee, S. et al. Mining biometric data to predict programmer expertise and task difficulty. *Cluster Computing 21*, 1 (2018), 1097–1107.

17. Lee, S. et al. Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis. In *Proceedings of the International Conference on Bioinformatics and Bioengineering (BIBE)*, IEEE, 2016, 350–355.

18. Mallow, J. et al. Superior memorizers employ different neural networks for encoding and recall. *Frontiers in Systems Neuroscience 9*, 128 (2015).

19. Moore, C. Neural mechanisms of expert skills in visual working memory. *Journal of Neuroscience 26*, 43 (2006), 11187–11196.

20. Nakagawa, T. et al. Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: A controlled experiment. In *Proceedings of the International Conference on Software Engineering (ICSE)*, ACM, (2014), 448–451.

21. Newell, A. *Unified Theories of Cognition.* Harvard University Press, 1994.

22. Peitek, N. et al. A look into programmers' heads. *IEEE Transactions on Software Engineering 46*, (2020), 442–462.

23. Peitek, N. et al. Simultaneous measurement of program comprehension with fMRI and eye tracking: A case study. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ACM, 2018, 1–24.

24. Poldrack, R. *New Mind Readers: What Neuroimaging Can and Cannot Reveal about our Thoughts*. Princeton University Press, 2018.

25. Siegmund, J. Program comprehension: Past, present, and future. In *Proceedings of the International Conference Analysis, Evolution, and Reengineering (SANER)*, IEEE, 2016, 13–20.

26. Siegmund, J. et al. Understanding source code with functional magnetic resonance imaging. In *Proceedings of the International Conference on Software Engineering (ICSE)*, ACM, 2014, 378–389.

27. Siegmund, J. et al. Measuring neural efficiency of program comprehension. In *Proceedings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, ACM, (2017), 140–150.

28. Yeh, M. et al. Detecting and comparing brain activity in short program comprehension using EEG. In *Proceedings of Frontiers in Education Conference*, IEEE, 2017), 1–5.

Authors

Janet Siegmund (siegmunj@fim.uni-passau.de) is a professor of computer science at Chemnitz University of Technology, Chemnitz, Germany.

Norman Peitek (norman.peitek@lin-magdeburg.de) is a Ph.D. student at the Leibniz Institute for Neurobiology, Magdeburg, Germany.

André Brechmann (brechmann@lin-magdeburg.de) is head of the Combinatorial NeuroImaging Core Facility at the Leibniz Institute for Neurobiology, Magdeburg, Germany.

Chris Parnin (cjparnin@ncsu.edu) is a professor of computer science at North Carolina State University, Raleigh, USA.

Sven Apel (sven.apel@acm.org) is a professor of computer science and chair of software engineering at Saarland Informatics Campus and Saarland University, Saarbrücken, Germany.