On the Influence of the Baseline in Neuroimaging Experiments on Program Comprehension

ANNABELLE BERGUM, Saarland University, Saarland Informatics Campus, Graduate School of Computer Science, Germany
NORMAN PEITEK, Saarland University, Saarland Informatics Campus, Germany
MAURICE REKRUT, German Research Center for Artificial Intelligence, Saarland Informatics Campus, Germany
JANET SIEGMUND, Chemnitz University of Technology, Germany
SVEN APEL, Saarland University, Saarland Informatics Campus, Germany

Background: Neuroimaging methods have been proved insightful in program-comprehension research. A key problem is that different baselines have been used in different experiments. A *baseline* is a task during which the "normal" brain activation is captured as a reference compared to the task of interest. Unfortunately, the influence of the choice of the baseline is still unclear.
Aims: We investigate whether and to what extent the selected baseline influences the results of neuroimaging experiments on program comprehension. This helps to understand the trade-offs in baseline selection with the ultimate goal of making the baseline selection informed and transparent.
Method: We have conducted a pre-registered program-comprehension study with 20 participants using multiple baselines (i.e., reading, calculations, problem solving, and cross-fixation). We monitored brain activation with a 64-channel electroencephalography (EEG) device. We compared how the different baselines affect the results regarding brain activation of program comprehension.
Results and Implications: We found significant differences in mental load across baselines suggesting that selecting a suitable baseline is critical. Our results show that a standard problem-solving task, operationalized by the Raven-Progressive Matrices, is a well-suited default baseline for program-comprehension studies. Our results highlight the need for carefully designing and selecting a baseline in program-comprehension studies.

CCS Concepts: • **General and reference** → **Empirical studies**; • **Human-centered computing** → *Empirical studies in HCI*.

Additional Key Words and Phrases: Program comprehension, EEG, eye-tracking

ACM Reference Format:
Annabelle Bergum, Norman Peitek, Maurice Rekrut, Janet Siegmund, and Sven Apel. 2025. On the Influence of the Baseline in Neuroimaging Experiments on Program Comprehension. *ACM Trans. Softw. Eng. Methodol.* 1, 1, Article 1 (January 2025), 26 pages. https://doi.org/10.1145/3744739

Authors' addresses: Annabelle Bergum, bergum@cs.uni-saarland.de, Saarland University, Saarland Informatics Campus, Graduate School of Computer Science, Germany; Norman Peitek, peitek@cs.uni-saarland.de, Saarland University, Saarland Informatics Campus, Germany; Maurice Rekrut, maurice.rekrut@dfki.de, German Research Center for Artificial Intelligence, Saarland Informatics Campus, Germany; Janet Siegmund, janet.siegmund@informatik.tu-chemnitz.de, Chemnitz University of Technology, Germany; Sven Apel, apel@cs.uni-saarland.de, Saarland University, Saarland Informatics Campus, Germany.

ACM 1049-331X/2025/1-ART1
https://doi.org/10.1145/3744739

1 INTRODUCTION

In the past decades, a growing number of studies on program comprehension have been conducted [71]. Clearly, the more we know about the cognitive processes involved in program comprehension, supporting tools and education of programmers can benefit, thereby improving productivity. A novel, growing set of studies use neuroimaging methods to locate and analyze activated brain areas and to measure the mental load during program comprehension [69]. These methods contribute to an objective measurement of how programmers understand source code. They offer an opportunity to create, test, and refine theories of program comprehension by obtaining insights into the cognitive processes [65], which led to several neuroimaging studies in software engineering [19, 31, 32, 43, 66]. Neuroimaging methods observe brain activation during a task of interest—comprehending source code, in our case. The challenge is that, typically, also nonspecific brain activation (e.g., associated with visual perception) occurs during the task of interest. Thus, the brain activation of a task cannot be analyzed directly, but is typically evaluated with respect to a selected *baseline*, as illustrated in Figure 1: By observing the *difference* in brain activation between a *task* (- - -) and a *baseline* (——), the task-specific brain activation (- - -) is extracted [44].

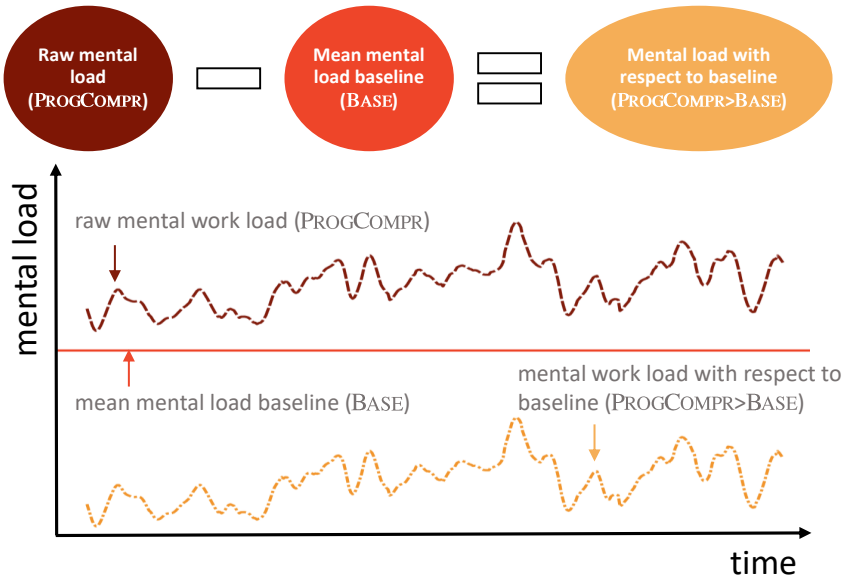


Fig. 1. Visualization of how the mental load (- - -) is computed from raw data (- - -) with respect to a baseline (——).

Needless to say, a suitable baseline is key to obtain comparable and valid experiment results. In functional magnetic resonance imaging (fMRI) experiments, the baseline is usually called *contrast condition*, whereas in EEG experiments, the term *baseline* is used. As our study provides insights for *all* neuroimaging methods, we will use the overarching term *baseline* throughout the paper.

As an example for the effect the choice of the baseline may have, consider the following two studies aiming at investigating which cognitive processes are involved in program comprehension, but using different baselines: Siegmund and others contrasted program comprehension with syntax error localization [63], whereas Ivanova and others contrasted program comprehension with the equivalent task, describing the code in natural language, using prose comprehension [32]. In these studies, the individual baselines were tailored to the respective study, while specific adjustments to a study increase internal validity but decrease the generalizability, as reflected in

the different results. While Siegmund and others found language-related brain areas to be active during program comprehension [63], Ivanova and others observed that the multi-demand system is largely active during program comprehension [32]. The two studies differ in several factors, including the programming language, code snippet selection, participant experience, but they also use different baselines. This makes a synthesis of the results inconceivable. Basically, we cannot pin down whether the differing results are due to the difference in selected baselines or other factors. In a growing field with a rising number of studies, this becomes increasingly problematic. We must understand the effect of baselines to obtain comparable study results and to design follow-up measures and experiments.

In this paper, we address this challenge by analyzing the influence of common baselines used in this area, and we provide insights into selecting a suitable baseline for a program-comprehension experiment. This way, we empower researchers to make an informed decision when planning experiments so that the methodological standard improves and results become more comparable. To this end, we have conducted a preregistered program-comprehension study with four baselines. We monitored the participants' brain activation with a 64-channel electroencephalography (EEG) device. We selected three baselines common in neuroimaging studies covering core competencies of program comprehension: reading a natural language text, calculating the value of an arithmetic expression, and problem solving as operationalized by the Raven-Progressive Matrices. Additionally, we include the current standard, that is, cross-fixation. In our analysis, we computed the mental load and the activation distribution over the brain during program comprehension with respect to the different baselines. This enables us to infer whether the choice of baseline influences the results. To obtain a detailed understanding of the baselines regarding their cognitive demands, we analyzed their differences and similarities to program comprehension based on the mental load, the activation distribution of the brain, and the reading strategy. Our results suggest problem solving as operationalized by Raven-Progressive Matrices is a well-suited default baseline for future program-comprehension studies, and study-specific needs require a targeted baseline choice (as discussed in Section 5.2).

In summary, we make the following contributions:

- A preregistered empirical study that examines the influence of four distinct baselines on the results of program-comprehension experiments;
- An analysis of differences and similarities in cognitive demands between program comprehension and the four baselines;
- A proposal of a standard problem-solving task operationalized by the Raven-Progressive Matrices, which is unrelated to programming, as a well-suited baseline for program-comprehension studies;
- A guideline for selecting suitable baselines for neuroimaging studies in the context of programming and related activities;
- A replication package with the complete study setup, data-analysis scripts, data, and the preregistration form, see Section 8.

2 METHODOLOGICAL CONSIDERATIONS FOR NEUROIMAGING STUDIES IN SOFTWARE ENGINEERING

Empirical experiments play a pivotal role in software-engineering research [70]. To ensure robust and reliable results across research teams, standards on how to properly design and conduct software-engineering experiments have been adopted and refined for two decades. For example, Kitchenham and others proposed a set of general experiment guidelines specifically for the field of software engineering, which includes experiment design, analysis, and reporting [37]. Based on this seminal work, more detailed guidelines on how to report the results of experiments have been proposed [34, 36]. More recently, researchers have reflected on the crucial role of specific properties of experiment designs, such as the inherent trade-off between internal and external validity [67].

2.1 Specifics of Neuroimaging Studies

With the advent of neuroimaging studies in software-engineering research, existing experiment guidelines must be refined further, because neuroimaging methods require a special experiment design regarding tasks and baselines. In particular, in neuroimaging experiments, we cannot directly assess the relevant brain activation during a task (i.e., a block of time in the experiment). Instead, we have to contrast the differences in brain activation between two tasks. We write "A > B" to indicate that we take task A, the cognitive process of interest (e.g., program comprehension), and subtract the brain activation that occurred in task B, the baseline (cf. Figure 1). If we obtain a positive value, the brain activation of the investigated cognitive process A is higher than that of the baseline B; if we obtain a negative value, it is lower in task A than in baseline B [39].

Finding a suitable baseline for a given task has been a widely debated topic in the general neuroscience literature [21]. While there is no universally suitable baseline, neuroscience researchers concluded that the baseline itself must be well understood. A commonly used baseline for neuroimaging experiments is *cross-fixation* [9, 49]. In a cross-fixation task, a participant is instructed to relax while focusing on a cross on a display to reduce muscle movements. Despite the popularity of cross-fixation, there are concerns about its suitability: Participants often state after an experiment session that "task-unrelated thoughts" appeared, so the relax time during cross-fixation does not imply neural inactivity [9]. That is, we do not know what these "task-unrelated thoughts" were, so it is unclear what cognitive processes took place. Furthermore, the cross-fixation can trigger different brain activation depending on the experiment [68].

2.2 Variety and Suitability of Baselines in Software-Engineering Research

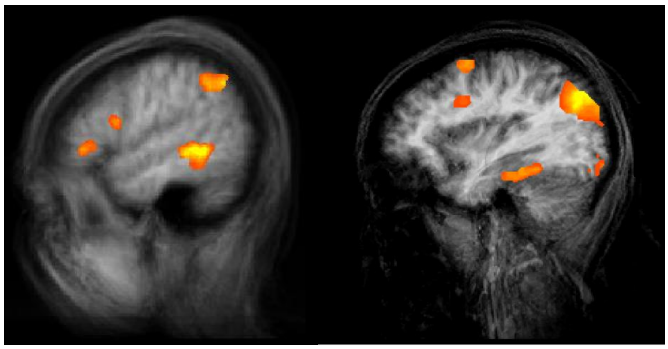
So far, neuroimaging studies in software engineering have used cross-fixation and, in addition, a substantial variety of further baselines, which we summarize along with their investigated tasks in Table 1. Even with a comparable operationalization of program comprehension across studies, the resulting brain activation might be completely different depending on the used baselines, as comparing the studies of Siegmund and others [63] and Ivanova and others suggests [32]. Another striking example is contrasting program comprehension with two different baselines in a single experiment: syntax error localization and an attention task [57]. Despite both analyzing brain-activation data on program comprehension, it reveals entirely different brain-activation patterns due to the use of different baselines (as illustrated in Figure 2) [51]. So, to understand the consequences of the baseline selection for experiment results and conclusions, we must not only understand the task itself, but also the baseline and its effect when contrasting it with the task. Furthermore, the differences may vary across individual participants, as participants' processing of tasks in such experiments can differ at different times across participants.

Selecting a suitable baseline is of particular importance for program comprehension studies. Program comprehension is a complex cognitive process involving multiple core competencies, including memory retrieval, logical reasoning, and language processing [63]. To unravel their roles, the ideal baseline is as close to the investigated task as possible, without capturing its "essence" [30].

A prototypical baseline for a neuroscience study in a block design (i.e., complex tasks that last, at least, 10 seconds) is either a different task with the same stimulus or the same task with a different stimulus [30]. The challenge in program comprehension is that it is an ensemble of different cognitive processes leading to high cognitive complexity. Depending on the focus of the major underlying cognitive process, there are different interpretations of what a suitable baseline for program comprehension is. For example, if we would contrast program comprehension with a simple working memory task, we would still observe high activation in the visual cortex (among other activations), which would not be considered essential of program comprehension. In essence, it is still unclear what a suitable baseline for program comprehension is and researchers must be careful when

Table 1. Overview of the variation of investigated tasks and baseline task(s) of neuroimaging studies in software engineering. Cross-fixation is part of all studies and is not included.

Study	Task under investigation	Baseline task(s)
Siegmund et al. [63]	Program comprehension	Syntax error localization
Siegmund et al. [66]	Program comprehension	Syntax error localization
Floyd et al. [19]	Program comprehension	Code review, prose review
Ivanova et al. [32]	Program comprehension	Prose comprehension
Peitek et al. [52]	Program comprehension	Bracket localization
Duraes et al. [16]	Bug localization	Program comprehension
Castelhano et al. [12]	Bug localization	Program comprehension
Huang et al. [29]	Data manipulation	Mental rotation
Krueger et al. [39]	Code writing	Prose writing
Hishikawa et al. [25]	Code execution, code completion, bug finding	Word counting
Ikutani et al. [31]	Code categorization	—
Huang et al. [28]	Code review	—



(a) Program Comprehension > Syntax (b) Program Comprehension > Attention

Fig. 2. Brain-activation example illustrating the influence of different baselines [51]. On the left (a), the brain activation of program comprehension was investigated with respect to a baseline of finding syntax errors, and on the right (b), with respect to a baseline of an attention task. Clearly, the resulting activated brain areas, highlighted in orange, are different, despite investigating the same cognitive process.

choosing the baseline of an experiment. But, there is no research that has investigated the effect of the baseline selection in program comprehension experiments, despite it being critically important.

2.3 Specificity and Generalizability during Baseline Selection: A Fundamental Trade-Off

Fundamentally, there is a trade-off between the specific needs of a particular experiment and the goal of generating knowledge in scientific research. As described above, neuroimaging experiments of program comprehension require carefully designed experiments, including the selection of a baseline, to address their specific research questions. For example, a study on comprehension of a visual programming language likely requires a contrast task that induces similar visual attention. This specialization is essential to collect insightful empirical data and draw meaningful conclusions. However, this specialization limits the generalizability of the results. When all

conducted experiments are custom-tailored, this severely limits how we can integrate and synthesize the results to the existing body of knowledge. This is problematic, as such synthesis is the core of science, rather than simply collecting individual results. Clearly, generalizing from individual neuroimaging experiments is crucial to fully unravel the cognitive processes of program comprehension.

Our research community must therefore aim at a balance between specificity and generalizability of designing neuroimaging experiments of program comprehension. While the customization of experiments to their specific needs is fully justified, we must also keep the needs of our broader scientific goals in sight. Our paper contributes to this goal by increasing our understanding of the effect of baselines in program-comprehension experiments, which is crucial for methodologically sound experiments.

3 STUDY METHODOLOGY

In this section, we describe our study design.

3.1 Research Goals

We formulate our research goal guided by the Goal-Question-Metric approach presented by Basili [5]. Our goal is to analyze the influence of the baseline and its similarity to program comprehension for the purpose of evaluation and identification of a suitable baseline. This endeavor is implemented with respect to program comprehension from the point of view of the programmer in the context of empirical software engineering. Our study addresses three research questions.

3.1.1 Influence of Baseline Selection (RQ_1). Numerous studies have aimed at answering questions about program comprehension using neuroimaging methods, often using different baselines, ranging from cross-fixation to reading prose (cf. Section 2.2 and Table 1). Understanding how different baselines affect results is necessary to create consistent studies that generate comparable results of brain-activation analysis.

> **RQ_1** To what extent does the baseline selection influence the results of brain-activation analysis?

The first neuroimaging studies on program comprehension have adapted experiment designs from neuroscience, but in different ways [19, 31, 63]. This led to a variety of task designs and baselines (cf. Table 1). If the choice of a baseline makes indeed a substantial difference, researchers are made aware of this. An answer to this research question would highlight the importance of an informed choice of the baseline and set a more solid foundation for future studies of program comprehension.

Operationalization. To answer RQ_1, we conduct an EEG study on program comprehension. Our choice of EEG is motivated by the fact that, with EEG, we can observe brain activity with a high temporal resolution [44]. This strength made EEG a standard measurement method of brain activation in assessing mental load [3]. We selected four different baselines: First, we include cross-fixation (CrossFix) as standard of neuroimaging studies and all neuroimaging studies of program comprehension, making sure we can produce comparable results. The other three baselines capture proclaimed core competencies of programmers: natural-language reading (Read), basic arithmetic calculations (Math), and problem solving operationalized by the Raven-Progressive Matrices (ProbSolv). These core competencies are also often the focus of studies that evaluate how these core competencies affect programming skill (acquisition) [2, 59]. Thus, the selection of baselines, while not complete, can be related to a wide agreed-upon understanding of program comprehension. To measure brain activation, we compute the mental load from the EEG signal (cf. Section 3.3.2). We chose mental load because it is an objective measure of the cognitive demands of a task, making it also the focus of many studies that investigate cognitive processes [3, 20, 26, 47, 72].

This operationalization allows us to test for significant differences between the obtained results on brain activation for program comprehension when using different baselines. The insights from RQ_1 lay the foundation for a methodological standard for future studies and aim at facilitating comparability across program-comprehension studies.

3.1.2 Cognitive Demands and Reading Strategy (RQ_2). Having evaluated to what extent different baselines affect experiment results, we dive deeper into how these baselines are different or similar in their cognitive demands to program comprehension. Our second research question is:

> **RQ_2** What differences and similarities in reading strategy and brain activation occur between program comprehension and the four baselines?

Reading strategy refers to a participant's chosen sequence of visual attention, as it forms the basis for subsequent cognitive processes during a task. Thus, answering this research question helps us to better understand the nature of the different baselines and disentangle the role of related cognitive processes on program comprehension, which is important to make informed decisions when designing neuroimaging experiments, providing further insights into the methodology for our field.

Operationalization. To answer RQ_2, we use a triangulation approach with the following aspects: First, we measure the mental load (cf. Section 3.3.2) of the different baselines in comparison to program comprehension. Second, we compare the four baselines in terms of activated brain areas, using *topoplots*, to show the spatial distribution of brain activation. Understanding *where* the brain activation occurs during the baseline tasks allows us to localize the source of mental load. Third, we investigate the reading strategy for each baseline in comparison with program comprehension. For this purpose, we use eye tracking, because eye movements differ depending on the reading strategy [7, 11]. This allows us to investigate our results from multiple angles and thus obtain more robust results. For example, a simple memory task is sufficient to induce high cognitive load, but would not be a well-suited contrast to program comprehension as its cognitive demands are entirely different. Thus, we also evaluate the similarity of the tasks by comparing participants' eye movements in addition to the mental load and its spatial distribution over the brain.

3.1.3 Toward a Default Baseline (RQ_3). Based on the results of RQ_1 and RQ_2, we search for a candidate for a well-suited "default" baseline for future program-comprehension studies. This is important, because, as the neuroimaging research branch in software engineering evolves, it spreads out to address more specific questions. Recently, researchers started to investigate program comprehension in more detail, for example, cognitive differences between experts and novices [31, 41, 53], the influence of code structures on top-down comprehension [66], or the visual presentation of source code [32]. The trend of evaluating the differences between very similar tasks (e.g., program comprehension of pretty-printed and disrupted code layout) leads to the observed difference in brain activation being small, likely too small, without a suitable baseline [66].

We illustrate this effect in Figure 3, where a well-suited baseline is closer to the mental load of program comprehension and thus helps to clearly reveal the differences between the two tasks. On a high level, we search for a default baseline providing a strong contrast in a general setting. More specifically, understanding the influence of the baseline also helps to choose the optimal one when the specific research goals require a custom-tailored baseline. Hence, we need a baseline that is as close as possible to the act of program comprehension to carve out the relative differences between potentially very similar program-comprehension tasks. This way, we can conduct studies to reliably pinpoint even small effects, such as the choice of identifiers [56] or indentation styles [6]. Thus, we formulate our third research question:

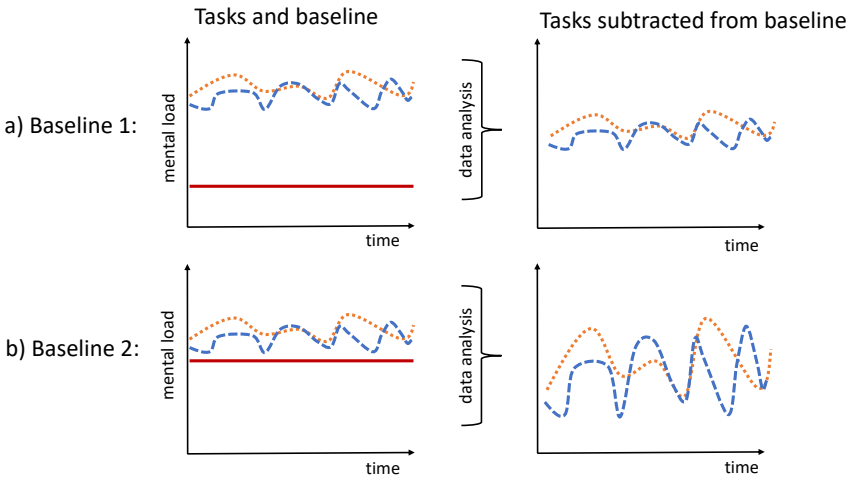


Fig. 3. Hypothetical example of two different baselines (——) and their effect on the results of an experiment (- - - and ·····). baseline 2, which is closer to the two investigated tasks in terms of mental load amplifies their differences during the analysis and is a better suited baseline as it allows to differentiate small effects.

> **RQ_3** Which baseline results in the best contrast and therefore qualifies as a well-suited default baseline for program-comprehension studies?

Operationalization. For RQ_3, we combine our findings from RQ_1 and RQ_2. On the one hand, we identify which baseline results in the most pronounced contrast between baseline and program comprehension. On the other hand, we investigate the cognitive demands for each baseline in terms of brain activation and reading strategy. In this study, we subtract the average mental load of the baseline task from the subsequent task, as presented in Figure 3. Ultimately, a suitable baseline yields only a small difference in brain activity when contrasted with program comprehension and also is similar in its cognitive demands.

3.2 Study Design and Conduct

To answer our research questions, we have conducted an EEG study, which we describe next. All materials, including snippets, tasks, and the experiment script, are available in an replication package (cf. Section 8).

3.2.1 Baseline Selection. When selecting the types of tasks for the baselines, we approached the choice from two angles. As the first study of its kind, we focused on non-programming tasks used in neuroscience studies to assess whether different baselines influence results. In a first step, since there is no comprehensive decomposition of the cognitive processes underlying program comprehension, we relied on our experience and insights from educational research [1, 17, 63]. We broke down program comprehension into several cognitive sub-processes: reading, calculating, abstracting, as these are plausible fundamental basics to understand program code according to the current understanding of the state of the art. Each of these cognitive processes is essential for program comprehension, and omitting even one of these would hinder understanding. However, these skills alone do not guarantee success in programming. There are many people who have mastered these three skills and are still unable to successfully program.

Neuroimaging studies have strict time constraints, which limits the number of baseline tasks that we can include in this experiment. In a second step, we reviewed fMRI studies on the described cognitive processes

to compare activated brain areas to those of program comprehension, as reported in the literature [55, 63, 66]. Activating the same brain areas indicates that the tasks share similar or the same processes at a neurological level, making them ideal baseline task candidates. In our analysis, we use Brodmann areas (BAs) as fundamental insights, which serve as an anatomical classification system in fMRI experiments. The entire brain is split into Brodmann areas based on cytoarchitectonic differences, which suggest that they serve different functional brain processes [10]. In what follows, we review a representative subset of the relevant literature.

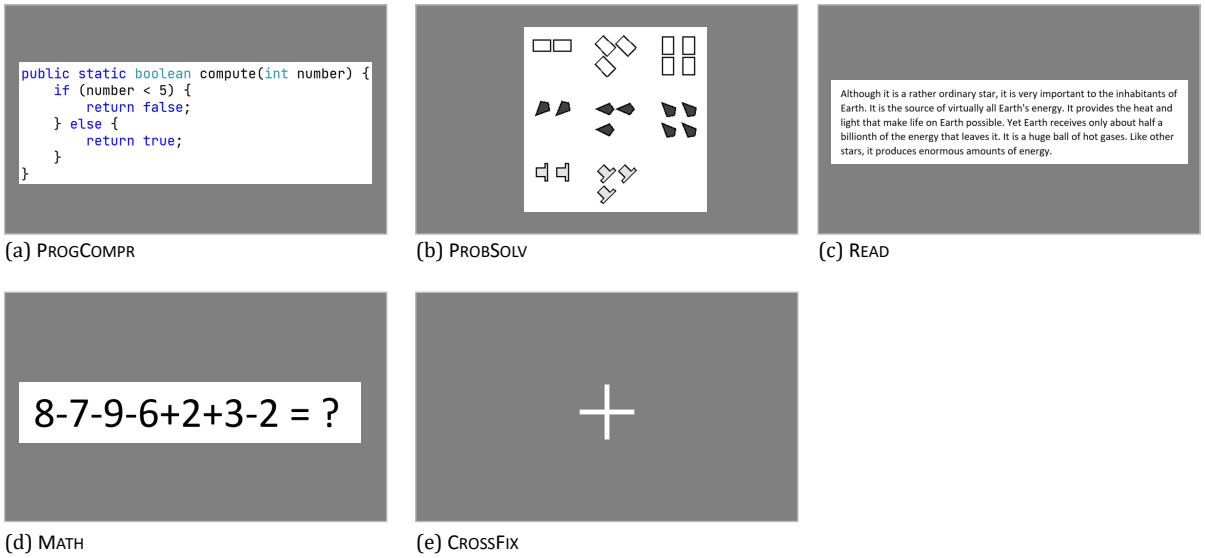
While fMRI studies on program comprehension are still fairly novel, there is still a good understanding which brain areas are relevant: BA 6, 21, 40, 44, and 47 are part of the brain network activated during program comprehension [55, 63, 66]. Price [60], Hagoort [22], and Hagoort and Indefrey [23] pinpoint the brain areas relevant to natural language processing to BA 6, 44, 45, and 47, which generally strongly overlap with the activated brain areas during program comprehension. In the area of mathematics, there is a lack of studies reporting detailed activation results in terms of Brodmann areas. Most notably, Newman and others [49] work with two different baselines (i.e., cross-fixation and verbalization of equation in every day context). They reported an activation in the BA 6, 7, and 40 (for cross-fixation) and 13, 32, and 40 (for verbalization). While not all, some of these brain areas are also activated during program comprehension. Finally, the term working memory has different interpretations in different fields. Here, we follow the review of Baddley [4], in which working memory is described as temporal memory with the ability to process data, which resembles the use in programming contexts, such that working memory allows persons to complete more complex tasks. According to Baddley, the central processing is supported by two components: First, the visuospatial sketchpad of activated BA 6, 19, 40, and 47, second, and the phonological loop of activated BA 40 and 44. These brain areas also show a partial overlap with the activation during program comprehension. Note that none of these sets of activated brain areas related to our candidates of related cognitive process are exhaustive. Nevertheless, they demonstrate that the intuitive skills necessary for program comprehension are also evident in the brain activation. Based on these considerations, we decided on using natural-language reading (Read), basic arithmetic calculations (Math), and problem solving (ProbSolv) as baseline tasks, contrasting program comprehension.

3.2.2 Task Selection. Next, we present our operationalization of the four baseline tasks as well as program comprehension. We explain the selected tasks in detail next and provide an overview in Table 2 as well as an illustration in Figure 4.

Table 2. Task Overview

Abbreviation	Task	Skill	Example
ProgCompr	Comprehend the functionality of a method without helpful naming	Program comprehension	Figure 4(a)
ProbSolv	Completing a Raven-Progressive-Matrix [61]	Instance of problem solving	Figure 4(b)
Read	Read a text about a searched term and determine the term	Natural-language reading	Figure 4(c)
Math	Calculating the sum of 7 single-digit numbers ($\in Z$)	Arithmetic calculation	Figure 4(d)
CrossFix	Looking at a fixation cross	Relax	Figure 4(e)

Program comprehension (ProgCompr). To operationalize program comprehension, we ask participants to comprehend code snippets in accordance with the state of the art in this area (see Table 1). Since we investigate



(a) ProgCompr (b) ProbSolv (c) Read

(d) Math (e) CrossFix

Fig. 4. Illustration of the program-comprehension task (a) and the four baseline tasks (b) to (e). All baselines and program comprehension tasks are available in the replication package.

the methodology of studies in this area, we must use a comparable comprehension task, which pre-defines the snippet complexity. Similar to prior studies, the snippets do not contain any beacons (i.e., variable names or method names hinting at the implemented functionality) to enforce bottom-up comprehension [58], which controls for confounding factors, such as domain knowledge or mental shortcuts [19, 52, 63, 66]. We selected Java as programming language due to its widespread use. In our study, we first show a snippet, and on a second page, a sample input with four possible outputs, one being correct. This way, we split program comprehension into two steps, so participants focus on comprehension first, not on computing an output or possibly switching between these two.

Natural-language reading (Read). During program comprehension, a programmer must also read and understand natural language, for example, the keywords `for` and `return`. In our study, we operationalize reading as in typical language studies by showing short texts about a specific term (e.g., sun) without using this term. In a second step, participants have to choose the term described in the text from four options.

Problem solving (ProbSolv). Problem solving is considered one important aspect of program comprehension. We operationalize problem solving with the well-established non-verbal Raven-Progressive Matrices test [61]. This is based on a general understanding of problem solving, which is not specific to programming, but includes deriving hypotheses and verifying them, and then applying the best hypothesis again. This high-level view covers a broad range of problem-solving applications. In a Raven-Progressive Matrix, the lower right corner misses one picture, and the participants need to select the missing picture from a set of four options presented on the next page.

Arithmetic calculation (Math). So far, the baselines covered natural-language processing and problem solving. The mathematical component of program comprehension is still missing. For mathematical calculation, we show

participants a chain of numbers with addition and subtraction operations, which they must mentally calculate. In a second step, participants must select the result from four options.

Cross-fixation (CrossFix). Finally, we included cross-fixation, since it is an established baseline [30]. It has no subsequent question, since participants are asked to relax and do nothing. However, this task as baseline has been critiqued, because uncontrollable, confounding "task-unrelated thoughts" can appear during this task, so the task does not imply neural inactivity (cf. Section 2.1).

3.2.3 Pilot Study. To evaluate whether the tasks that we selected are suitable for our study, we conducted an online pilot study. The tasks of the pilot study had to be completed by the participants (i) within a reasonable time and (ii) with a high rate of correct answers: If completed too fast, we obtain insufficient or low-quality data for this condition[1]; if too slow, we lose valuable time in the experiment. A high rate of correct answers is important, so that we can be sure the intended cognitive process actually takes place.

Most of the 120 participants were university students that were selected for the pilot study with an intermediate level of programming experience (cf. Table 3) [15, 46]. The demographics and programming experience of the participants were self-reported, and the correctness and the response time of the baseline and program-comprehension tasks were recorded. Based on these data, we selected the final set of tasks for each baseline and program comprehension for the actual EEG study (see Section 3.2.4).

Table 3. Demographics of the pilot study

Demographics/Experience measures	# / Mean ± SD
Participants	120
Age (in years)	23.7 ± 3.42
Years at university	3.58 ± 2.88
Years of programming	5.54 ± 3.72
Years of Java programming	3.21 ± 3.04

For each baseline, we chose the 7 tasks with the highest correctness rate and a mean response time between 10 and 60 seconds. We planned with 28 baseline tasks and 28 program-comprehension tasks with an expected overall time of 55 minutes (based on our pilot study). For program comprehension, we needed 28 tasks. From the pilot study, we were able to include 13 program-comprehension tasks and 7 simplified tasks, as their original version had a too high average response time or a too low correctness rate. Additionally, we added 8 new tasks from other studies on program comprehension[2] to meet the number of required program-comprehension tasks. We provide all data, including answer times, correctness values, and criteria, in the replication package.

3.2.4 Experiment Plan. Having selected the tasks for the baselines and program comprehension, we could conduct the actual study. We display the tasks one after another, starting with a baseline task, followed by a program-comprehension task, and so on, with small rest periods without a task in between (cf. Figure 5). These rest periods are necessary to let the activation return to a pre-task level. One hour is the usual time limit to avoid fatigue effects in neuroimaging experiments [19, 52, 53, 66]. We pseudo-randomize the task order, minimizing biases of learning effects and missing data. The ordering of the baselines stays the same within a participant, but is pseudo-randomized between participants. After one-third and two-thirds of the overall tasks, we encourage participants to take a longer break (5 minutes). We end the experiment after a maximum of 60 minutes. The

[1] because the EEG analysis used in the main study needs a few seconds of data to compute the mental load
[2] based on https://www.tu-chemnitz.de/informatik/ST/CodeWebsite/

CrossFix baseline is set to 20 seconds. The rest periods are set to 10 seconds between baseline and program comprehension and 20 seconds between program comprehension and baseline. From our experience, break times can be perceived as more annoying than relaxing for some participants, so we give participants the option to continue after the minimum break necessary for high data quality (i.e., 50% of the scheduled time). We set a maximum time for the baseline tasks of 180 seconds for the question and 60 seconds for the answer (but they were never exceeded in the experiment).

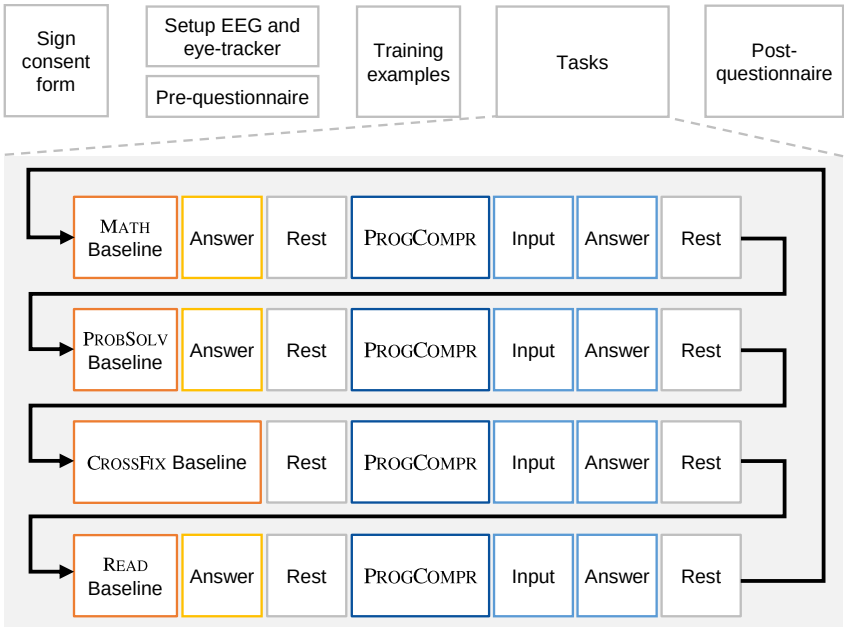


Fig. 5. Overview of the study and the task order. The order of the baselines (Math, ProbSolv, CrossFix, Read) was randomized across participants.

We use the questionnaire of Peitek and others [53] to collect demographic data and assess programming experience (which is based on a validated questionnaire by Siegmund and others [64]). We include the demographics and programming-experience questionnaire, as well as the post-questionnaire in the replication package, see Section 8. Our study was approved by the ethical review board of the faculty of Mathematics and Computer Science at Saarland University (Application number: 22-06-4).

3.2.5 Experiment Execution and Data Collection. Overall, the experiment lasted around 2 hours for each participant. We welcomed the participants, and they signed the consent forms. Then, they completed the demographics and programming-experience questionnaires and clicked through the instructional presentation while the EEG was placed on their head, which usually takes 30 min. The participants were encouraged by the instructor to ask clarification questions. After that, we calibrated the eye-tracker, and subsequently, the tasks started. At the end of the experiment, we removed the EEG cap and the participants answered the questions of the post-questionnaire in a semi-structured interview. Finally, we paid a compensation of 10 euros. No participant reported discomfort.

The EEG lab is located in a room with reduced light and minimal distractions, such as external noise or mobile devices. Participants sat in a comfortable chair minimizing unnecessary muscle movement to avoid noise and

artifacts in the EEG signal. We recorded the EEG signals using *LiveAmp 64*[3], a 64-channel EEG system. We used a sampling rate of 500 Hz and the international 10–20 system of electrode placement [33], to cover the entire scalp and obtain spatial information from the brain recordings. We used the Tobii Pro Fusion eye-tracker[4] with a sampling rate of 250 Hz, attached to the screen to answer RQ_2. We presented the stimuli with a PsychoPy script (version v.2021.2.3 [50], also available in the replication package).

3.2.6 Participants. We recruited participants at Saarland University via e-mail lists and online bulletin boards. We asked for basic knowledge of Java or a related programming language, such as C#, since the code snippets did not use any advanced Java-specific constructs.

We provide an overview of our participants' demographics in Table 4, which is in line with other neuroimaging studies. Our participant sample consists of largely university students with an intermediate level of programming experience [15, 46]. Overall, we invited 23 participants. We excluded the data of one because of unreadable data files due to a technical error. We also excluded two complete participant data sets due to their behavioral data consisting of a majority of tasks being classified as outliers (cf. Section 3.3.1), leaving us with 20 participants in the analyzed data set.

Table 4. Overview of participant demographics of our EEG study and a subset of (4 out of 49) measures from our programming-experience questionnaire.

Demographics/Experience measures	# / Mean ± SD
Number of (included) participants	20
Female	5
Male	15
Age (in years)	23.7 ± 2.56
Years of learning programming	5.4 ± 3.36
Years of professional programming	2.0 ± 0.92
Years of Java programming	2.9 ± 3.23
Number of known programming languages	5.1 ± 1.39

3.3 Analysis Protocol

Next, we describe the protocol of analyzing the collected data. We largely reuse the processing pipeline of Peitek and others [53]. The results are presented in Section 4.

3.3.1 Outlier Removal. We start the analysis by removing outliers in task completion time. Specifically, we exclude all data points[5] where the response option "Skip" was selected, the response time was outside 1.5 times the interquartile range, or a response time faster than 3 seconds, which is the minimum time to calculate the mental load. This way, data points where participants did not thoroughly attempt to solve the task were removed. This led to the removal of more than half of the initial data points for two (slow) participants, so we excluded their entire data sets from the study.

In total, we obtained 17 usable data sets. Since we aimed at 20 participants, as this is a common participant number in our field [69], we invited 3 additional participants.

[3]Brain Products GmbH, https://brainvision.com/products/liveamp-64/
[4]Tobii AB, https://www.tobiipro.com/product-listing/fusion/
[5]A data point here is a tuple of the response times of a participant for a baseline task and the following program-comprehension task.

We further excluded 75 data points: 33 skipped or not completed, 41 with a response time lying outside the ≥ 1.5 interquartile range, and one with a response time $\leq 3s$. This leaves us with 20 participants with a total of 485 data points.

3.3.2 EEG Data.

Data Cleaning. As a standard procedure in this area, we removed noise from the EEG data—particularly motion artifacts—using Hamming-filtered finite impulse response (FIR) filters. Then, we removed power line noise using a notch filter with a lower cutoff frequency of 49 and an upper cutoff frequency of 51 Hertz. Additionally, for more robust performance in subsequent analysis, high-frequency noise was removed using a bandpass filter in the range of 4 to 200 Hertz [62].

For one participant, three channels (23, 24, 56) were probably broken, because the values of these three channels were extremely high. Since we did not observe such high values for any other channels for any other participants, we excluded the three channels from the analysis of this participant. Given the type of experiment, which requires participants to observe different parts of the screen, it is impossible to avoid eye movements and other muscle activities. To avoid bias during the interpretation of the EEG signal, we used blind source separation to reduce such artifacts via an automated implementation of independent component analysis (ICA) of the EEGLAB toolbox [14][6] (again following Peitek and others [53]).

Mental Load. We based our assessment of mental load on a spectral analysis, using the different frequency bands of the EEG signal. For robustness, we calculate the mental load using two different approaches, following Peitek and others [53]: the theta/alpha power ratio and the relative power analysis.

For the first approach, we calculate mental load as *theta/alpha power ratio*, according to Holm and others [26] and Kartali and others [35]. We use a sliding window of 3 seconds and a shift of 0.1 seconds. We average the power spectral density to obtain the mean power for each frequency band of interest. The measure of mental load is calculated by dividing the mean theta value by the mean alpha value within a window. This way, we can observe the time course of mental load during a task. For calculating the specific mental load of program comprehension (cf. Figure 1), we subtract the average mental load of the baseline task from the mental load of the subsequent program-comprehension task [44].

The second approach to measure the mental load is *relative power analysis* per band and per channel along the alpha (8–12Hz), beta (12–30Hz), gamma (30–45Hz), and theta (4–8Hz) bands, following Lee and others [41]. This approach is sensitive to the spatial location of brain activation by taking the channel of the electrode into account. The resulting *topoplots* provide us with a coarse-grained map of brain activation, indicating the possible sources of mental load, as we discussed in Section 3.1.2.

Statistical Significance Testing: Program Comprehension. Since the base activation of the brain shifts throughout the experiment [42], program comprehension is only evaluated with respect to its directly preceding baseline task (not, for example, with respect to a baseline task from 20 minutes ago). This leads to grouping all brain activation measurements during program comprehension by the preceding baseline, such that in ProgCompr>Read only all ProgCompr tasks that had a preceding Read task are considered.

To ensure that this grouping does not influence the program-comprehension task, we tested for significant differences in mental load between program-comprehension tasks (ProgCompr after Math vs. ProgCompr after Read vs. ProgCompr after ProbSolv vs. ProgCompr after CrossFix). Since the mental-load data were not normally distributed (Shapiro-Wilk test: $p \leq 0.03$), we tested for statistical significance with a Kruskal-Wallis test [40]. It revealed no significant differences ($p = 0.569$), indicating that the four groups of program-comprehension tasks did not show significant differences in mental load (which is good, as they are the same task).

[6]version 2021.1, https://sccn.ucsd.edu/eeglab/

Statistical Significance Testing: Between Baselines. To evaluate whether there are significant differences between the mental load of different baselines, we also used a Kruskal-Wallis. If significant, we conducted a post-hoc pair-wise comparison with a Mann-Whitney U test [45] with the standard $p < 0.05$ criteria, completed with Benjamini-Hochberg false discovery rate [8] to correct multiple testing. We calculated the effect size using Cliff's delta [13].

3.3.3 Eye-Tracking Data. We qualitatively analyze the eye-tracking data for RQ_2 in terms of gaze patterns and frequently fixated areas in the stimuli [27]. We preprocess the eye-tracking data by identifying fixations and saccades from the raw stream of coordinates. To this end, we use `I2CM`, a noise-robust algorithm [24] for fixation classification. We plot the fixations and saccades as scan paths to visualize participants' reading strategy [27].

4 RESULTS

After applying the analysis protocol to our data, we can answer our three research questions in this section.

4.1 RQ_1: Influence of Baseline Selection

We calculated the mental load with respect to each baseline (ProgCompr>Read, etc.). The data are not normally distributed (Shapiro-Wilk: $W > 0.93$, $p < 0.031$ for each). A Kruskal-Wallis test showed significant differences ($X^2 = 20.90$, $p \leq 0.001$). The pair-wise Mann-Whitney-U tests revealed that some pairs differ significantly (see Table 5) after applying the Benjamini-Hochberg procedure to adjust the false discovery rate (FDR). We visualize the mental load over time in Figure 6, which shows differences in mental load based on the selected baseline. There is a strong contrast between ProgCompr analyzed with respect to CrossFix (·····), on the one hand, and ProgCompr analyzed with respect to ProbSolv (- - -), on the other, such that the differences in mental load significantly affect the analysis.

Table 5. RQ_1: Significant differences in mental load when contrasting program comprehension with the four baselines. We analyzed the baselines pairwise. Significant results are marked in bold.

Pairwise Different Baselines		Mann-Whitney U	FDR	Cliff's delta
ProgCompr > CrossFix	vs. ProgCompr > Math	**0.027**	0.040	−0.163 (small)
ProgCompr > CrossFix	vs. ProgCompr > Read	**0.002**	**0.004**	0.231 (small)
ProgCompr > CrossFix	vs. ProgCompr > ProbSolv	**<0.001**	**<0.001**	−0.323 (small)
ProgCompr > Math	vs. ProgCompr > Read	0.300	0.300	—
ProgCompr > Math	vs. ProgCompr > ProbSolv	**0.019**	**0.039**	0.177 (small)
ProgCompr > Read	vs. ProgCompr > ProbSolv	0.103	0.124	—

Thus, we answer our first research question:

> **RQ_1** The choice of the baseline has a statistically significant influence on the results of the brain-activation analysis of program comprehension.

4.2 RQ_2: Cognitive Demands and Reading Strategy

For a more comprehensive understanding of the similarities and differences of the cognitive demands of the baselines to program comprehension, we first describe the three aspects (mental load, reading strategy, brain activation) of the triangulation individually before combining them into one view.

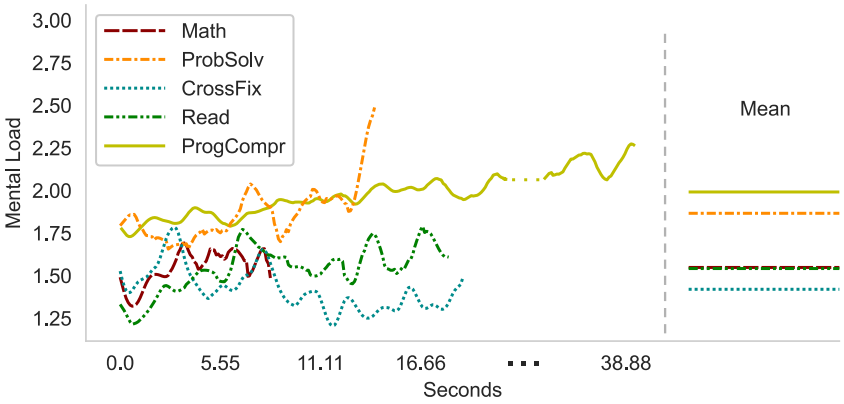


Fig. 6. RQ_2: Average of the mental load of all baselines and program comprehension over time. The data are plotted until >50% of the participants have completed the task, to ensure that the average remains meaningful. Thus, the individual lines are of different lengths, which represent their mean answer time. On the right side, their overall means are shown, for a better comparison.

Mental Load. Considering the mean mental load over time, shown in Figure 6, we find that Math (- - -) was demanding only for a short amount of time. For ProbSolv (-·-·-), Read (-··-··), and ProgCompr (——), we observed a longer demand on mental load, but Read and CrossFix (·····) stayed on a lower level. Overall, the mental load of ProbSolv most closely resembles the mental load of program comprehension. This is backed up by statistical tests: Since the data are not normally distributed (Shapiro-Wilk: $W > 0.88$, $p \leq 0.001$), we applied a Kruskal-Wallis test. We found significant differences between the tasks ($X^2 = 28.80$, p 0.00 ≤ 0.05). A post-hoc Mann-Whitney U test found statistical significant differences for ProgCompr vs. Math, Read and CrossFix (each: $U \geq 8033$, $p < 0.014$, adjusted p with FDR < 0.034). The effect sizes are small (Cliff's delta: 0.19, 0.19, and 0.25, respectively), though. There is no statistical significant difference, with only a negligible effect size between ProgCompr vs. ProbSolv.

Reading Strategy. Regarding the reading strategy, we analyzed the scan paths obtained from the eye-tracking data. We show examples of scan paths across the baselines and program comprehension in Figure 7.[7]

The example in Figure 7 is representative of a general pattern: Math and Read both show a linear reading strategy from left to right, top to bottom, with only a few fixations on previously seen parts. The number of fixations is fairly equally spread across the stimuli. ProbSolv and ProgCompr are both non-linear: participants are first overviewing the stimuli and then quickly focus on relevant points of interest. For both tasks, there are many refixations on previously seen parts. Since participants focus on specific parts, the fixations are not equally distributed over the stimuli, but clustered in certain areas of interest.

In general, the eye-tracking data point to problem solving as operationalized by the Raven-Progressive Matrices as being most similar to program comprehension in terms of reading strategy.

Brain Activation. In terms of activated brain areas, we analyze the topoplots in Figure 8. Across all program comprehension tasks (A), we find largely comparable brain activation across the entire brain, indicating that, at a more detailed level, the effect of the preceding baseline task on the activation of program comprehension cannot be observed (which is as expected). However, when we analyze the activation of program comprehension with

[7]We omit CrossFix here, as it does not trigger a reading strategy.

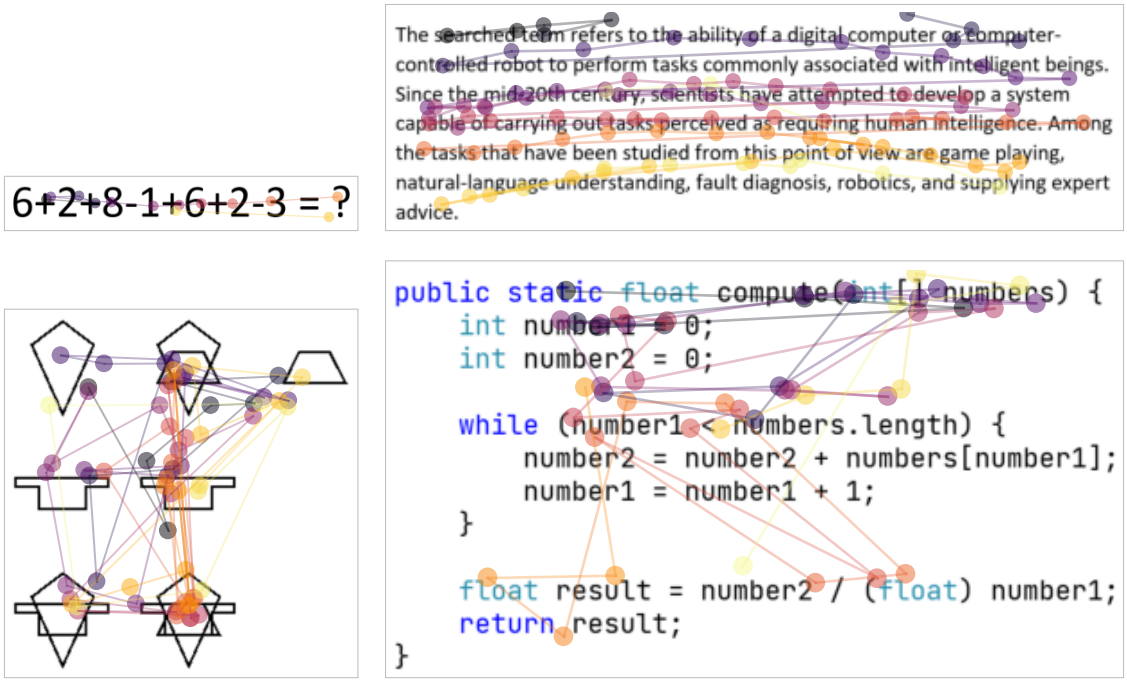


Fig. 7. RQ_2: Representative scan-path examples of a participant across three baselines (Math, Read, ProbSolv) and ProgCompr. Math and Read show a linear reading strategy. ProbSolv and ProgCompr show a non-linear reading strategy.

respect to the different baselines (B), differences in reading strategy are also visible in the brain activation: In CrossFix and Math, the activation in the front of the brain, mainly caused by eye movement, is showing red topoplot areas. That is, the eye-movement-induced activation was not sufficiently canceled out, whereas it is almost completely canceled out in Read and ProbSolv.

Overall, we answer our second research question by triangulating all three findings, where ProbSolv always is most similar to program comprehension:

RQ_2	Problem solving as operationalized by the Raven-Progressive Matrices most closely resembles program comprehension considering all three aspects (mental load, reading strategy, and brain activation). Reading, calculation, and cross-fixation are substantially different from program comprehension in terms of their cognitive demands.

4.3 RQ_3: Toward a Default Baseline

Based on the insights of RQ_2, problem solving seems well-suited as a default baseline for program-comprehension studies. To be a good fit, it must also show an average mental load close to the average mental load of program comprehension, which would allow us to have a more detailed view of program comprehension (cf. Figure 3). In Figure 6, we visualize the mental load of the four baselines and the averaged mental load of program comprehension. Program comprehension and problem solving (ProbSolv) both trigger a comparably high mental load (≥ 1.8) compared to the other baselines (≤ 1.6). Since the observed mental load of ProbSolv is closest to

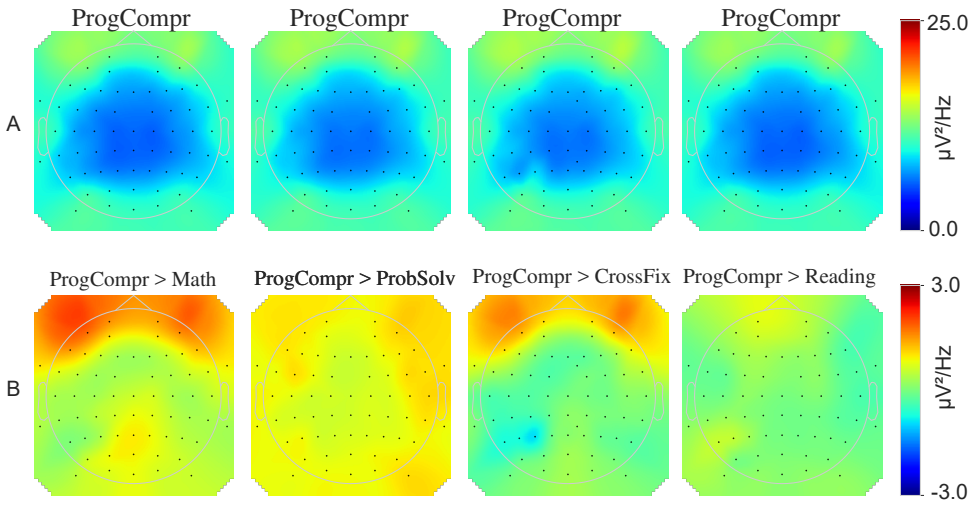


Fig. 8. RQ$_2$: Topoplots for program comprehension contrasted with all four baselines. **A**: Average of all frequency bands for ProgCompr grouped by the preceding baseline. **B**: Average of all frequency bands for ProgCompr contrasted with the four baselines. The yellow to red areas are activated and the blue areas are deactivated.

the mental load of program comprehension, it qualifies as a candidate for a suitable default baseline for future studies in this area.

RQ$_3$	Problem solving (ProbSolv) appears to be a well-suited default baseline for program-comprehension studies.

4.4 Exploration of Post-Questionnaire Data

To obtain a comprehensive picture of the selected tasks and their relation to program comprehension, we interpret the participants' feedback and comments directly following the EEG study. This helps us to better differentiate our findings on how and why these tasks evoke different mental work load and different reading strategies. The post-questionnaire data are available in the replication package.

4.4.1 Difficulty and Similarity of Baseline Tasks and Program Comprehension. We aimed at finding a baseline task that is as close to program comprehension as possible, without capturing its "essence" [30]. Therefore, we are interested how participants perceived the difficulty of the baseline tasks and their similarity to program comprehension. It is particularly important that the baseline tasks are of similar difficulty to program comprehension to generate sufficiently high baseline activation for comparison. Thus, in addition to the subjective rating, we explored the open-text feedback on the perceived similarity, to not miss potentially relevant details.

Overview. Our collected subjective data, shows that many participants viewed ProbSolv as similarly to slightly more difficult than program comprehension, whereas all other baselines were considered as (much) easier (see Figure 10), which is in line with our mental load analysis in Section 3.1.2. On a high level, participants perceived three baseline tasks as rather similar to program comprehension, but not CrossFix (see Figure 9).

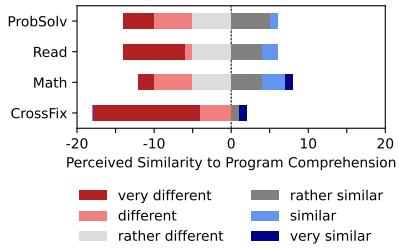


Fig. 9. The perceived similarity of baselines to program comprehension differs across participants, except for the cross-fixation, which shows a strong shift to the *not similar* direction.

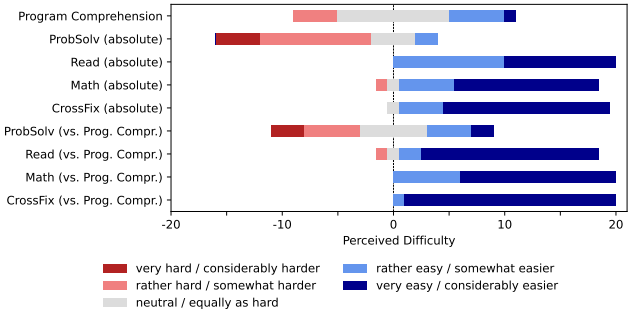


Fig. 10. The difficulty of problem solving is mostly perceived as equally or harder as the other baselines and program comprehension; the other baselines are perceived as easier than program comprehension.

Math: Overall, participants described the arithmetic problems as easier, as they require less memorization than program comprehension, which is consistent with the mental load analysis in Section 3.1.2. Participants further reported that the approach and skills are different, while some participants viewed arithmetic problems and program comprehension to be similar, since program comprehension also involves arithmetic.

Read: Furthermore, participants reported that natural language reading was easy for several reasons. First, the correct answer was sometimes clear before the answer options or even after the first sentence, which then led to skipping the remaining text. Second, they also described reading as more subconscious, automated, and less reflective than program comprehension, which also aligns with our results in the mental load analysis. They further expressed that program comprehension requires specialized knowledge and, unlike the task Read, does not allow an educated guess. Participants also acknowledged that natural language reading and program comprehension both require reading and that they used similar comprehension strategies to search a solution.

ProbSolv: Participants perceived problem solving as rather difficult, since the result was not clear unless the answer options were shown; thus they had to memorize the abstracted concepts. They also expressed that difficulty varied widely among tasks. The participants explained that both, problem solving and program comprehension, are based on recognizing patterns, structures, and sequences. For both tasks, they used logical thinking and noted that prior knowledge being useful. Specifically, experience with algorithms was helpful, which they did not have for the problem-solving tasks.

CrossFix: Finally, participants perceived the cross-fixation as very different and very simple, since nothing needed to be done. Some participants stated staying motionless was more challenging here than during cognitively demanding tasks. Furthermore participants explained that it is difficult to relax in a study setting, and they disliked the interruption of the workflow.

ProgCompr: Overall, the participants assessed program comprehension as more difficult than the other baseline tasks, but overall straightforward. They mentioned the recursion and the lack of a specific input number as difficult, as this required more memorization. They noted further that the answer was more clear compared to the ProbSolv task, but they needed to jump back and forth more often and needed to reread parts, which was not the case in the Read and Math tasks.

Summary. The subjective rating provided insights explaining the higher similarity of program comprehension and problem solving. Specifically relevant appears to be the amount of information (e.g., numbers) to be kept in

working memory and how common and, therefore, intuitively automatic the tasks are (e.g., natural language reading vs. problem solving). Specifically, the higher working memory load of problem solving can be explained that the participants needed to keep more in mind compared to natural language reading or during the arithmetic problems.

> The post-study feedback, providing us with an additional dimension of insights, matches our results based on the mental load analysis and eye-tracking analysis as presented in Section 3.1.2. Overall, we conclude that our prior finding that problem solving can serve as a reasonable default baseline is also supported by the subjective perception of our participants.

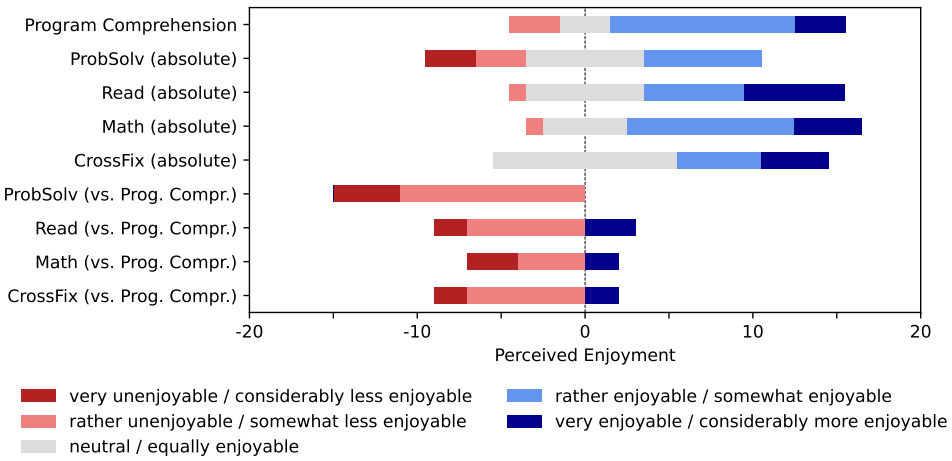


Fig. 11. Compared to program comprehension, the tasks appear rather unenjoyable. Especially the problem solving task rating is shifted in the unenjoyable direction.

4.4.2 Task Enjoyment of Baseline Tasks and Program Comprehension. To collect high-quality data, one aspect is to keep participants actively engaged throughout the experiment. We tried to create an enjoyable experience within the constraints of our research design and assessed participants' perceived enjoyment of all tasks. Almost all participants rated the program-comprehension tasks as more enjoyable than any of the baselines (see Figure 11).

Based on the open text feedback we learned, that the difficulty and success of the tasks influenced participants' enjoyment: Cognitively demanding tasks (such as problem solving or program comprehension) were perceived as exhausting, but also as challenging providing a sense of reward when solved successfully. We recommend including a "None of these" answer option to mitigate potential negative thoughts of participants when their believed answer option was not a choice. We found a clear tendency for program comprehension to be the most enjoyable task. Given that participants volunteered for a program comprehension study, this is to be expected. Cross-fixation was viewed negatively, since it was boring and tiring to do nothing. Therefore, minimizing downtime during the experiment is crucial for participant engagement while preserving the option to take breaks.

> Challenging tasks can be enjoyable when the solution is within reach of the participants' abilities. The downtime can be omitted by self-regulated break durations, as the study design allows.

4.4.3 Open-Text Feedback on Task-Unrelated Thoughts. Finally, we examined participants' statements on their task-unrelated thoughts. Ideally, we aim to minimize such thoughts, since mind wandering reduces data quality, as we measure these thoughts rather than the investigated tasks.

Interestingly, participants (17/20) reported that cross-fixation triggered a large number of task-unrelated thoughts in our experiment, considerably more than the other baselines (Math: 1/20, Read: 6/20, ProbSolv: 3/20), but in line with prior literature reporting task-unrelated thoughts [9].

> Participants have task-unrelated thoughts during downtime. Thus, designing engaging (but not overwhelmingly difficult) tasks and minimizing opportunities for distractions is important to collect high-quality data.

5 DISCUSSION

Having presented our results, we now discuss their implications in two directions: the influence of the choice of the baseline and the search for a suitable default baseline for program comprehension.

5.1 Influence of Baseline Selection (RQ_1)

Neuroscience research has shown that the choice of a suitable baseline is a crucial part of the experiment design [21]. Our study adds to this for the area of program comprehension. The results of RQ_1 show that using different baselines for program comprehension significantly influence the experiment results. Consequently, it is imperative to carefully consider this influence as a factor when comparing and synthesizing existing research, as two studies with different baselines will obtain different, possibly contradicting results. While our study focused on EEG, our insights apply to all research that relies on a baseline, including program comprehension studies using fMRI and functional near-infrared spectroscopy (fNIRS) [18], which we will discuss in more detail below.

5.2 Suitable Baseline for Studies (RQ_2 and RQ_3)

After showing that different baselines affect the results of program-comprehension experiments, we investigated which baseline is the most suitable for future studies. This is necessary for a solid methodological foundation to move toward valid and comparable studies.

The results of RQ_2 and RQ_3 suggest that problem solving in the form of a Raven-Progressive-Matrix task is a suitable default baseline for studies of program comprehension. A triangulation with eye tracking providing corroborating evidence, highlighting that problem solving triggers a non-linear reading behavior and a cognitive demand similar to program comprehension. Reading and calculation trigger linear reading behavior and lower cognitive demand and thus do not fulfill the criteria for a suitable baseline in a default case for studies on program comprehension (without specific requirements due to the study-specific research questions, cf. Section 2.2). Finally, our exploration of the post-questionnaire regarding the participants' subjective views further corroborate these findings. Specifically, they provide qualitative insights into how problem solving engages cognitive processes that are more related to those of program comprehension than the other baselines.

Guided by these findings, future research can make an informed decision on which baseline to select. Our proposal of problem solving as default baseline for EEG studies of program comprehension provides a common foundation for this research area.

As mentioned above, it is important to note that our study is only the start of an empirical evaluation of baselines. For example, the reading strategy of programmers differs depending on experience, such that novices tend to read source code more linearly than experienced programmers [11, 54]. That is, for a study specifically targeting novice programmers, reading or calculations could be a suitable baseline with respect to a similar linear, left-to-right reading strategy; however, novice programmers showed higher cognitive load during program

comprehension [53], which could disqualify less demanding baselines such as reading or calculations. This shows how complex selecting a well-suited baseline, depending on the study demands, is. For example, other cognitive processes of interest in software engineering (e.g., writing source code) may not be ideal for using problem solving as baseline. Future studies must carefully consider designing and selecting the suitable baseline that ideally fits to the research goals. In general, future research shall continue to investigate suitable baselines for this and other specific research questions. We provide a foundation for this endeavor with our experiment design, which is publicly available in our replication package, see Section 8.

Baseline for Other Constructs. Clearly, software-engineering studies that have special requirements regarding the baseline will presumably not resort to our default baseline for program comprehension and shall carefully investigate their choice of a baseline. Nevertheless, based on our provided data and experiment design, other researchers can adapt the experimental protocol and baseline definitions for similar cognitive constructs, so that the software engineering research community can develop and agree upon standards for researching cognitive constructs with neuroimaging methods. We would also like to point out that it is possible to use two baselines in an experiment. For example, researchers could design their experiment with a custom baseline for the specialized needs for their research goals, and our general-purpose baseline of RPMs to ensure comparability to other studies. However, this does reduce the number of possible tasks in a time-limited neuroimaging experiment.

Baseline for fMRI and fNIRS. We conducted our study with EEG, since it has several advantages over fMRI or fNIRS beyond the lower cost and better device availability. fMRI studies limit the time of the participants in the scanner to around 30 minutes (e.g., [63]) and require longer rest periods between tasks for the BOLD response to return to normal. In an EEG experiment, we are not only able to shorten rest periods, but also let participants solve tasks for around 45 minutes without running into fatigue effects. This substantial increase in the number of tasks per participant is quite relevant, as our investigation into baselines is novel and there was no prior work we could rely on to design a targeted experiment with only two or three baselines. We included several baselines, which required a more time for each participant, and therefore EEG was the better choice without having to conduct multiple fMRI experiments.

While we concentrated on EEG, many insights apply to other neuroimaging methods. fMRI and fNIRS examine the activated brain areas in more spatial detail than EEG, and therefore have slightly different requirements that must be considered: While our EEG data consider mental load only, fMRI and fNIRS collect three-dimensional data of brain activation. For fMRI studies, we therefore require a baseline that is not only reliable across various participants and a good reference point in one dimension, but also a good fit in 3-dimensional brain activation patterns. The results of our study, specifically the spatial distribution in the topoplots and the reading strategy based on eye tracking, indicate that ProbSolv is a suitable baseline for fMRI and fNIRS, as well. This issue requires attention in future studies: Is the brain activation of a problem-solving task (ProbSolv) measured with fMRI or fNIRS sufficiently similar to be used as a baseline for program comprehension? A replication study with fMRI shall be conducted to evaluate whether we can confirm our insights. Our EEG study with its experiment design lays the foundation for such follow-up experiments, such as a targeted fMRI study with the most promising baseline candidates.

6 THREATS TO VALIDITY

In this section, we discuss threats to construct, internal, and external validity of our study. We minimized potential threats to validity by carefully designing the experiment with an extensive pilot study. We have further preregistered our study at the Open Science Foundation, which means we could not change or add research questions with hypotheses after the data were gathered. We committed to the evaluation details before starting data collection, to prevent post-hoc “fishing for results”.

6.1 Construct Validity

The mental-load measure may deviate from the *actual* mental load. We mitigated this threat by using an established mental-load measure for EEG [26, 35], which also has been tested specifically for use in software engineering [48]. In the same vein, our operationalization of program comprehension was inspired by similar experiments [19, 38, 53, 63]. Furthermore, we analyzed all of the baselines in a single study to avoid confounding factors such as interpersonal variation.

6.2 Internal Validity

Regarding experiment design, we pseudo-randomized the task order to counteract learning and fatigue effects over the time of the experiment. We minimized variance in participant instruction by showing an instructional presentation rather than the experimenter explaining the experiment.

A substantial threat to validity arises from our selection of specific tasks for each category (e.g., which snippets to show for program comprehension). We mitigated this threat by collecting the tasks from previous studies in the field and by conducting a pilot study to carefully select the specific tasks.

6.3 External Validity

Our study included 20 participants, which is in line with neuroimaging experiments in software engineering [69]. Due to the repeated-measures experiment design, we collected a sufficient amount of data, increasing reliability. Our participants reported experience measures compared with the majority of neuroimaging studies in our field. Regarding our participant sample, our participants were intermediate programmers, and therefore the results are neither specific to high-performing programmers nor novices. However, participants with different profiles (e.g., expert programmers) may experience different mental loads for different tasks. Future research shall further test our results across a wide variety of programmers.

7 CONCLUSION

Baselines are a fundamentally important element in neuroimaging studies. In an EEG study on program comprehension with different baselines, we have shown that the baseline has a significant effect on the results of neuroimaging experiments on program comprehension. Our results challenge the community to be careful when adapting neuroimaging methods to software engineering. While promising, we are advised to properly modify the methods for our needs. With this study, we take a crucial step in this direction by showing the importance of the baseline and initiating a search for a suitable baseline and advocate for reporting the used baseline. In particular, we found that problem solving in the form of a Raven-Progressive-Matrix is a suitable candidate for a default baseline for future studies on program comprehension, if the specific research setting does not require otherwise. Leveraging a common baseline, or adding this baseline as an additional evaluation option, will enable us to work together towards a deeper understanding of program comprehension. Future research shall investigate further baselines, such as related tasks to program comprehension or tasks with varying difficulty. Furthermore, we shall test the adaption to other neuroimaging methods, for which a good starting point would be fMRI, but also fNIRS and MEG (magnetoencephalography) are of interest. Our study lays the foundations for this endeavor.

8 DATA AVAILABILITY

Along with this submission, we provide an online repository[8] (i.e., replication package, pilot-study data, scripts for analysis protocol, and additional figures, EEG and eye tracking data) along with our preregistration on osf.io[9].

[8] https://github.com/brains-on-code/baseline_eeg_study
[9] https://osf.io/eh245/?view_only=d5adb6163267436783ba9e58f9304b46

ACKNOWLEDGMENT

We thank all participants of our study. Furthermore, we thank Yannick Lehmen for their technical support during data acquisition. This work has been supported by the European Union as part of ERC Advanced Grant "Brains On Code" (101052182) and by the German Research Foundation through Collaborative Research Center TRR 248 – CPEC (389792660). Rekrut's work is supported by the German Federal Ministry of Education and Research (01IW20003).

REFERENCES

[1] Ana Paula Ambrósio, Fábio Moreira Costa, Leandro Almeida, Amanda Franco, and Joaquim Macedo. 2011. Identifying cognitive abilities to improve CS1 outcome. In *2011 Frontiers in Education Conference (FIE)*. IEEE, F3G–1.

[2] Ana Paula Ambrosio, Leandro da Silva Almeida, Joaquim Macedo, and Amanda Franco. 2014. Exploring Core Cognitive Skills of Computational Thinking. In *Annual Conf. Psychology of Programming Interest Group (PPIG)*. 25–34.

[3] Pavlo Antonenko, Fred Paas, Roland Grabner, and Tamara Van Gog. 2010. Using Electroencephalography to Measure Cognitive Load. *Educational Psychology Review* 22, 4 (2010), 425–438.

[4] Alan Baddeley. 2001. Is Working Memory Still Working? *The American Psychologist* 56, 11 (2001), 851–864.

[5] Victor Basili. 1992. *Software Modeling and Measurement: The Goal/Question/Metric Paradigm.* Technical Report CS-TR-2956 (UMIACS-TR-92-96). University of Maryland at College Park.

[6] Jennifer Bauer, Janet Siegmund, Norman Peitek, Johannes Hofmeister, and Sven Apel. 2019. Indentation: Simply a Matter of Style or Support for Program Comprehension? In *Proc. Int'l Conf. Program Comprehension (ICPC)*. ACM, 11.

[7] Roman Bednarik. 2012. Expertise-Dependent Visual Attention Strategies Develop over Time during Debugging with Multiple Code Representations. *Int'l Journal of Human-Computer Studies* 70, 2 (2012), 143–155.

[8] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)* 57, 1 (1995), 289–300.

[9] Jeffrey Binder, Julia Frost, Thomas Hammeke, Patrick Bellgowan, Stephen Rao, and Robert Cox. 1999. Conceptual Processing During the Conscious Resting State: A Functional MRI Study. *Journal of Cognitive Neuroscience* 11, 1 (1999), 80–93.

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

[11] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye Movements in Code Reading: Relaxing the Linear Order. In *Proc. Int'l Conf. Program Comprehension (ICPC)*. IEEE, 255–265.

[12] João Castelhano, Isabel Duarte, Carlos Ferreira, João Durães, Henrique Madeira, and Miguel Castelo-Branco. 2019. The Role of the Insula in Intuitive Expert Bug Detection in Computer Code: An fMRI Study. *Brain Imaging and Behavior* 13, 3 (2019), 623–637.

[13] Norman Cliff. 1993. Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychological Bulletin* 114, 3 (1993), 494–509.

[14] Arnaud Delorme and Scott Makeig. 2004. EEGLAB: An Open Source Toolbox for Analysis of Single-Trial EEG Dynamics Including Independent Component Analysis. *Journal of Neuroscience Methods* 134, 1 (2004), 9–21.

[15] Hubert Dreyfus and Stuart Dreyfus. 1986. *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer.*

[16] João Durães, Henrique Madeira, João Castelhano, Isabel Duarte, and Miguel Castelo-Branco. 2016. WAP: Understanding the Brain at Software Debugging. In *Proc. Int. Symposium Software Reliability Engineering (ISSRE)*. IEEE, 87–92.

[17] Madeline Endres, Madison Fansher, Priti Shah, and Westley Weimer. 2021. To Read or to Rotate? Comparing the Effects of Technical Reading Training and Spatial Skills Training on Novice Programming Ability. In *Proc. Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 754–766.

[18] Sarah Fakhoury, Devjeet Roy, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope. 2020. Measuring the Impact of Lexical and Structural Inconsistencies on Developers? Cognitive Load during Bug Localization. *Empirical Softw. Eng.* 25 (2020), 2140–2178.

[19] 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'l Conf. Software Engineering (ICSE)*. IEEE, 175–186.

[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'l Conf. Software Engineering (ICSE)*. ACM, 402–413.

[21] Debra Gusnard and Marcus Raichle. 2001. Searching for a Baseline: Functional Imaging and the Resting Human Brain. *Nature Reviews Neuroscience* 2, 10 (2001), 685–694.

[22] Peter Hagoort. 2014. Nodes and Networks in the Neural Architecture for Language: Broca's Region and Beyond. *Current Opinion in Neurobiology* 28 (2014), 136–141.

[23] Peter Hagoort and Peter Indefrey. 2014. The Neurobiology of Language Beyond Single Words. *Annual Review of Neuroscience* 37 (2014), 347–362.

[24] Roy Hessels, Diederick Niehorster, Chantal Kemner, and Ignace Hooge. 2017. Noise-Robust Fixation Detection in Eye Movement Data: Identification by Two-Means Clustering (I2MC). *Behavior Research Methods* 49, 5 (2017), 1802–1823.

[25] Kenji Hishikawa, Kenji Yoshinaga, Hiroki Togo, Takeshi Hongo, and Takashi Hanakawa. 2023. Changes in Functional Brain Activity Patterns Associated with Computer Programming Learning in Novices. *Brain Structure and Function* 228, 7 (2023), 1691–1701.
[26] Anu Holm, Kristian Lukander, Jussi Korpela, Mikael Sallinen, and Kiti Müller. 2009. Estimating Brain Load from the EEG. *The Scientific World Journal* 9 (2009), 639–651.
[27] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. OUP Oxford.
[28] Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. 2020. Biases and Differences in Code Review Using Medical Imaging and Eye-Tracking: Genders, Humans, and Machines. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 456–468.
[29] Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. 2019. Distilling Neural Representations of Data Structure Manipulation using fMRI and fNIRS. In *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE, 396–407.
[30] Scott Huettel, Allen Song, and Gregory McCarthy. 2014. *Functional Magnetic Resonance Imaging*. Vol. 3. Sinauer Associates.
[31] Yoshiharu Ikutani, Takatomi Kubo, Satoshi Nishida, Hideaki Hata, Kenichi Matsumoto, Kazushi Ikeda, and Shinji Nishimoto. 2021. Expert Programmers have Fine-Tuned Cortical Representations of Source Code. *Eneuro* 8, 1 (2021), 1–16.
[32] Anna Ivanova, Shashank Srikant, Yotaro Sueoka, Hope Kean, Riva Dhamala, Una-May O'reilly, Marina Bers, and Evelina Fedorenko. 2020. Comprehension of Computer Code Relies Primarily on Domain-General Executive Resources. *eLife* 9:e58906 (2020), 1–24.
[33] Herbert Jasper. 1958. Report of the Committee on Methods of Clinical Examination in Electroencephalography. *Electroencephalogr. Clin. Neurophysiol.* 10 (1958), 370–375.
[34] Andreas Jedlitschka, Marcus Ciolkowski, and Dietmar Pfahl. 2008. Reporting Experiments in Software Engineering. In *Guide to Advanced Empirical Software Engineering*. Springer, 201–228.
[35] Aneta Kartali, Milica Janković, Ivan Gligorijević, Pavle Mijović, Bogdan Mijović, and Maria Leva. 2019. Real-Time Mental Workload Estimation Using EEG. In *Human Mental Workload: Models and Applications*. Springer International Publishing, 20–34.
[36] Barbara Kitchenham, Hiyam Al-Khilidar, Muhammed Babar, Mike Berry, Karl Cox, Jacky Keung, Felicia Kurniawati, Mark Staples, He Zhang, and Liming Zhu. 2008. Evaluating Guidelines for Reporting Empirical Software Engineering Studies. *Empirical Softw. Eng.* 13, 1 (2008), 97–121.
[37] Barbara Kitchenham, Shari Pfleeger, Lesley Pickard, Peter Jones, David Hoaglin, Khaled El Emam, and Jarrett Rosenberg. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *Transactions on Software Engineering (TSE)* 28, 8 (2002), 721–734.
[38] Makrina Kosti, Kostas Georgiadis, Dimitrios Adamos, Nikos Laskaris, Diomidis Spinellis, and Lefteris Angelis. 2018. Towards an Affordable Brain Computer Interface for the Assessment of Programmers' Mental Workload. *Int.Í Journal of Human-Computer Studies* 115 (2018), 52–66.
[39] Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological Divide: An fMRI Study of Prose and Code Writing. In *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE, 678–690.
[40] William Kruskal and Allen Wallis. 1952. Use of Ranks in One-Criterion Variance Analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621.
[41] 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'l Conf. on Bioinformatics and Bioengineering (BIBE)*. IEEE, 350–355.
[42] Ren Li, Jared Johansen, Hamad Ahmed, Thomas Ilyevsky, Ronnie Wilbur, Hari Bharadwaj, and Jeffrey Siskind. 2018. Training on the Test Set? An Analysis of Spampinato et al. *arXiv preprint arXiv:1812.07697* (2018), 1–18.
[43] Yun-Fei Liu, Judy Kim, Colin Wilson, and Marina Bedny. 2020. Computer Code Comprehension Shares Neural Resources with Formal Logical Inference in the Fronto-Parietal Network. *Elife* 9 (2020), e59340.
[44] Steven Luck. 2014. *An Introduction to the Event-Related Potential Technique*. MIT press, 251–253.
[45] Henry Mann and Donald Whitney. 1947. On a Test of whether One of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* (1947), 50–60.
[46] Jerry Mead, Simon Gray, John Hamer, Richard James, Juha Sorva, Caroline Clair, and Lynda Thomas. 2006. A Cognitive Approach to Identifying Measurable Milestones for Programming Skill Acquisition. *ACM SIGCSE Bulletin* 38, 4 (2006), 182–194.
[47] Julio Medeiros, Ricardo Couceiro, João Castelhano, Castelo Branco, Gonçalo Duarte, Catarina Duarte, João Durães, Henrique Madeira, Paulo Carvalho, and César Teixeira. 2019. Software Code Complexity Assessment Using EEG Features. In *Proc. Int'l Conf. Engineering in Medicine and Biology Society (EMBC)*. IEEE, 1413–1416.
[48] Júlio Medeiros, Ricardo Couceiro, Gonçalo Duarte, João Durães, João Castelhano, Catarina Duarte, Miguel Castelo-Branco, Henrique Madeira, Paulo de Carvalho, and César Teixeira. 2021. Can EEG Be Adopted as a Neuroscience Reference for Assessing Software Programmers' Cognitive Load? *Sensors* 21, 7 (2021), 2338.
[49] Sharlene D. Newman, Gregory Willoughby, and Benjamin Pruce. 2011. The Effect of Problem Structure on Problem-Solving: An fMRI Study of Word Versus Number Problems. *Brain Research* 1410 (2011), 77–88.
[50] Jonathan Peirce, Jeremy Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, Erik Kastman, and Jonas Lindeløv. 2019. PsychoPy2: Experiments in Behavior Made Easy. *Behavior Research Methods* 51, 1 (2019), 195–203.

[51] Norman Peitek. 2022. *A Neuro-Cognitive Perspective of Program Comprehension.* Ph. D. Dissertation. University of Technology Chemnitz.
[52] Norman Peitek, Sven Apel, Chris Parnin, André Brechmann, and Janet Siegmund. 2021. Program Comprehension and Code Complexity Metrics: An fMRI Study. In *Proc. Int'l Conf. Software Engineering (ICSE)*. ACM, 524–536.
[53] Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. 2022. Correlates of Programmer Efficacy and their Link to Experience: A Combined EEG and Eye-Tracking Study. In *Proc. Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 120–131.
[54] Norman Peitek, Janet Siegmund, and Sven Apel. 2020. What Drives the Reading Order of Programmers? An Eye Tracking Study. In *Proc. Int'l Conf. Program Comprehension (ICPC)*. ACM, 342–353.
[55] Norman Peitek, Janet Siegmund, Sven Apel, Christian Kästner, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2020. A Look into Programmers' Heads. *IEEE Trans. Softw. Eng.* 46, 4 (2020), 442–462.
[56] Norman Peitek, Janet Siegmund, and André Brechmann. 2017. Enhancing fMRI Studies of Program Comprehension with Eye-Tracking. In *Proc. Int'l Workshop on Eye Movements in Programming (EMIP)*. Freie Universität Berlin, 22–23.
[57] Norman Peitek, Janet Siegmund, Chris Parnin, Sven Apel, Johannes Hofmeister, and André Brechmann. 2018. Simultaneous Measurement of Program Comprehension with fMRI and Eye Tracking: A Case Study. In *Proc. Int'l Symposium Empirical Software Engineering and Measurement (ESEM)*. ACM, 24:1—-24:10.
[58] Nancy Pennington. 1987. Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs. *Cognitive Psychology* 19, 3 (1987), 295–341.
[59] Chantal Prat, Tara Madhyastha, Malayka Mottarella, and Chu-Hsuan Kuo. 2020. Relating Natural Language Aptitude to Individual Differences in Learning Programming Languages. *Scientific Reports* 10 (2020), 3817.
[60] Cathy Price. 2012. A Review and Synthesis of the First 20 Years of PET and fMRI Studies of Heard Speech, Spoken Language and Reading. *NeuroImage* 62, 2 (2012), 816–847.
[61] John Raven. 2000. The Raven's Progressive Matrices: Change and Stability over Culture and Time. *Cognitive Psychology* 41, 1 (2000), 1–48.
[62] Maurice Rekrut, Mansi Sharma, Matthias Schmitt, Jan Alexandersson, and Antonio Krüger. 2020. Decoding Semantic Categories from EEG Activity in Object-Based Decision Tasks. In *Proc. Int'l Winter Conf. Brain-Computer Interface (BCI)*. IEEE, 1–7.
[63] 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'l Conf. Software Engineering (ICSE)*. ACM, 378–389.
[64] Janet Siegmund, Christian Kästner, Jörg Liebig, Sven Apel, and Stefan Hanenberg. 2014. Measuring and Modeling Programming Experience. *Empirical Softw. Eng.* 19, 5 (2014), 1299–1334.
[65] Janet Siegmund, Norman Peitek, André Brechmann, Chris Parnin, and Sven Apel. 2020. Studying Programming in the Neuroage: Just a Crazy Idea? *Commun. ACM* 63, 6 (2020), 30–34.
[66] 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 *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 140–150.
[67] Janet Siegmund, Norbert Siegmund, and Sven Apel. 2015. Views on Internal and External Validity in Empirical Software Engineering. In *Proc. Int'l Conf. Software Engineering (ICSE)*, Vol. 1. IEEE, 9–19.
[68] Craig Stark and Larry Squire. 2001. When Zero is Not Zero: The Problem of Ambiguous Baseline Conditions in fMRI. *Proc. National Academy of Sciences* 98, 22 (2001), 12760–12766.
[69] Barbara Weber, Thomas Fischer, and René Riedl. 2021. Brain and Autonomic Nervous System Activity Measurement in Software Engineering: A Systematic Literature Review. *Journal of Systems and Software* 178 (2021), 110946.
[70] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers.
[71] Marvin Wyrich, Justus Bogner, and Stefan Wagner. 2023. 40 Years of Designing Code Comprehension Experiments: Systematic Mapping Study. *Comput. Surveys* 56, 4 (2023), 1–42.
[72] Martin Yeh, Dan Gopstein, Yu Yan, and Yanyan Zhuang. 2017. Detecting and Comparing Brain Activity in Short Program Comprehension Using EEG. In *Frontiers in Education Conference*. IEEE, 1–5.