Master's Thesis

Using ACT-R to predict Mental Workload during Code Comprehension

Yannick Lehmen

September 29, 2025

Advisors: Dr. Marvin Wyrich
Anna-Maria Maurer
Annabelle Bergum

Reviewers: Prof. Dr. Sven Apel
Prof. Dr. Vera Demberg

Chair of Software Engineering ◇ Saarland Informatics Campus ◇ Saarland University

Erklärung Statement

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne die Beteiligung dritter Personen verfasst habe, und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht. Insbesondere bestätige ich hiermit, dass ich bei der Erstellung der nachfolgenden Arbeit mittels künstlicher Intelligenz betriebene Software (z. B. ChatGPT) ausschließlich zur Textüberarbeitung/-korrektur und zur Code-Vervollständigung und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. Alle mittels künstlicher Intelligenz betriebenen Software (z. B. ChatGPT) generierten und/oder bearbeiteten Teile der Arbeit wurden kenntlich gemacht und als Hilfsmittel angegeben. Ich erkläre mich damit einverstanden, dass die Arbeit mittels eines Plagiatsprogrammes auf die Nutzung einer solchen Software überprüft wird. Mir ist bewusst, dass der Verstoß gegen diese Versicherung zum Nichtbestehen der Prüfung bis hin zum Verlust des Prüfungsanspruchs führen kann.

I hereby declare that I have written this thesis independently and without the involvement of third parties, and that I have used no sources or aids other than those indicated. All passages taken directly or indirectly from publications or other external sources have been identified as such. In particular, I confirm that I have used AI-based software (e.g., ChatGPT) exclusively for the following permitted sub-tasks: text rewriting/revision and code completion, and not to address or formulate the main research questions of the thesis. All parts of the thesis that were generated and/or edited using AI-based software (e.g., ChatGPT) have been disclosed and documented in accordance with the documentation requirements. I agree that the thesis may be checked using plagiarism detection software, including checks for the use of such software. I am aware that any violation of this declaration may result in failing the examination and lead to losing the right to be examined.

Saarbrücken, _____ _____
(Datum Date) (Unterschrift Signature)

Einverständniserklärung (optional) Declaration of Consent (optional)

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____
(Datum Date) (Unterschrift Signature)

Abstract

Identifying complex code early in the development process is highly beneficial in software engineering, because complex code is more difficult to understand, and costlier to maintain and extend. Current research focuses on the use of code complexity metrics and neuro-imaging techniques to measure code complexity. However, while both approaches have advantages, research also highlights some important issues that occur when using them. Code complexity metrics often correlate only weakly with human performance, while neuro-imaging based measures require costly set-ups and a person must be actively working on a task for the workload to be measured. As such, these measures are not scalable. Additionally, individual differences in the measurements make them hard to generalise, further restricting their practical usage.

In this thesis, we present an alternative approach to measuring code complexity. We used an existing model of code comprehension in the cognitive architecture Adaptive Control of Thought-Rational (ACT-R) to simulate code comprehension and predict mental workload during the process. With ACT-R simulating human cognition, it could provide a way to incorporate mental workload findings in a scalable and automated complexity measurement. To evaluate our approach, we conducted a code comprehension study with 34 participants, in which we measured subjective workload using the NASA-TLX questionnaire. In addition, we collected EEG-based workload measurements in the form of the theta-to-alpha ratio as a band power based measurement and the mean P300 amplitude as an event-related-potential based measurement.

We found that the predicted workload is linearly related to both subjective workload and the theta-to-alpha ratio. Although we observed that the mean P300 amplitude was higher for more difficult snippets, we found no significant correlation with the predicted workload. This suggests that the ACT-R model fails to capture an aspect of workload represented in the event-related potentials. Based on a line-by-line analysis of the workload prediction and EEG-based workload measurements, we concluded that the model's simplified simulation of calculations is the most likely cause of this.

Overall, we show that the prediction of both subjective and EEG-based workload measurements in program comprehension tasks is possible, and that the workload prediction can be a valuable tool in improving future ACT-R models.

Contents

List of Figures

Acronyms

EEG electroencephalography
fMRI functional magnetic resonance imaging
ACT-R Adaptive Control of Thought - Rational
fNIRS functional Near-Infrared Spectroscopy
ERP Event-Related Potentials
BOLD blood-oxygenation-level-dependent
NASA-TLX NASA Task Load Index
FDR False discovery rate
IDE Integrated Development Environment
QN Queueing Network
OEU Overall Expected Utilization
QN-ACTR Queueing Network - Adaptive Control of Thought Rational
APT ACT-R Programming Tutor

1 Introduction

1.1 Goal of this Thesis

Program comprehension is an important aspect of the work of software developers, with many tasks consisting to a large extent of programmers reading and understanding code that is often written by another person. Examples of such tasks are code maintenance [77] and the extension of existing code [78]. More complex code is harder to understand, and therefore maintaining it is less productive [6, 7]. The complexity of a program is not just dependent on what it does but also on how it is implemented, as even semantically equivalent code can vary in complexity depending on the exact implementation [26]. As such, it would be helpful to be able to detect overly complex code early. This would allow developers to rewrite complex code to be easier to understand, or add explanation to code that cannot be simplified. Future work on the respective code would then benefit from the reduced complexity, making maintenance easier and more productive. In addition to potential advantages in industry, understanding what causes code to be hard to understand could potentially help with teaching new developers by spending additional time on those areas.
One existing way to judge code complexity is by using questionnaires, such as the NASA Task Load Index (NASA-TLX) [33], which are designed to have a user judge the difficulty of performing a certain task after having performed that task. This subjective difficulty rating correlates strongly with performance on programming tasks [65]. However, the practical use of subjective workload measures is limited, because they require developers to interrupt their work and can only be used after a task is already finished. As such, this approach is largely confined to studies and is not commonly used in industry.
An alternative to subjective difficulty rating are complexity metrics. These metrics are computed automatically, and do not require significant interruption in the workflow of a developer. Examples of such metrics include Halstead's Difficulty metric [51], McCabe's Cyclomatic Complexity [52] and Lines of Code [22]. Despite their widespread use, these metrics have many problems. They correlate only weakly with subjective difficulty and behavioural data [65, 71] and most metrics are not empirically validated regarding their correlation with software quality. [57]. Thus, relying on code complexity metrics is insufficient for analysing code complexity.
More recently, studies have used neuro-imaging techniques based on tools such as functional magnetic resonance imaging (fMRI), electroencephalography (EEG), or eye-tracking as a measure of mental workload in program comprehension [74]. While these measurements show equivalent or better correlation with the perceived difficulty of a comprehension task than

traditional complexity metrics [53, 65], they have problems in practical use because they require external devices the developer needs to wear while working, and can only evaluate the workload during work, not predict it beforehand.

Given these issues of neuro-imaging techniques in practical usage it would be helpful to find a way to make use of their advantages in a way that can be applied automatically, independent of individual programmers, and without interfering in a programmer's workflow. One possible approach to do so would be to develop models that predict mental workload during code comprehension. These models would be created using data from neuro-imaging techniques, but could be applied to the program code without any additional measurements, in much the same way as traditional code complexity metrics are. Then, the models could predict which parts of the code are likely to induce a high mental workload. This knowledge could then be used in development, without requiring expensive and difficult measurements to be conducted first. In this thesis, we explore one approach to construct such a model using a cognitive architecture, a family of models from the field of neuropsychology that are intended to simulate human cognition. The architecture we use is ACT-R [4], with us making use of an already existing model developed in this architecture to simulate code comprehension in simple programs [18]. ACT-R is a rules-based model, simulating human behaviour in a task by following rules laid out by the modeller. While ACT-R itself only provides direct insight into the time and correctness of the simulated data, it is possible to use the trace of internal activity in the model to predict the mental workload during a task as was done by an algorithm developed by Jo et al. [39]. In a code comprehension setting, this means that it is possible, assuming a model that correctly simulates human behaviour on a comprehension task, to pinpoint what part of the program code causes difficulty to arise and potentially even why exactly that difficulty causes errors. In this way, it would be possible to pinpoint specific lines or sections of code that are likely to cause excessive mental workload or errors in code comprehension, as opposed to code complexity metrics, which usually only operate on a level of entire methods or modules.

1.2 Contributions

We investigate the usability of the ACT-R model by Closheim [18] for the purpose of predicting mental workload during the comprehension of simple source code snippets. To this end, we make the following contributions:

- We present an EEG study of code comprehension with 32 participants, explain how we collected subjective and neurological measures of mental workload, and simulate the same experiment with the ACT-R model developed by Closheim. We show that the model, with additional fine-tuning, provides an acceptable simulation of behavioural data.
- We demonstrate that the existing ACT-R-based workload prediction algorithm by Jo et al. can be applied to program comprehension tasks to predict subjective workload data. Additionally, we show that the algorithm can be used to predict EEG-based workload measurements, both for entire snippets and single lines. This is, to our

knowledge, a novel use case of this algorithm, which so far has only been applied to larger categories of tasks in psychological experiments and has not been evaluated for the prediction of EEG-based workload measurements.

- We investigate the use of ERP and the P300 component in the ERP as a measurement of mental workload in program comprehension. We find that, while the P300 is of some use for this purpose, there is a larger effect of workload later in the ERP that peaks at around 650 ms. We show an effect of the growing problem size of calculations on this effect, describe a component that might cause it, and identify three possible identities (P600, late frontal positivity, or slow positive wave) of this component in existing research.
- We propose possible future changes to the ACT-R model by Closheim based on discrepancies between the ACT-R model and behavioural data and between the predicted workload and workload measures (both subjective and EEG-based) in different snippets and lines of code. These changes could improve how well the model aligns with human behaviour and mental processes.

1.3 Overview

The remainder of this thesis is divided into five chapters. In Chapter 2, we introduce the background of this thesis. We present the current state of code comprehension research, and the role that mental workload and neuro-imaging techniques play in it. We provide a basic introduction into the cognitive architecture ACT-R, show an algorithm that can be used to predict workload based on an ACT-R model and present an existing ACT-R model of code comprehension. In Chapter 3, we introduce our research questions and the experiment we conduct to answer them. Afterwards, we present our results and discuss their implications in Chapter 4. In Chapter 5, we present related work dealing with the intersections of code comprehension, ACT-R models and workload measurements. In the last chapter we summarise our findings and present possible future work.

2

Background

In the following, we present important information for this thesis. We begin by explaining topics that are necessary to understand the general idea of our proposal, including the current state of code comprehension research, the general concept of mental workload, and some neuro-imaging techniques. Afterwards, we present the ACT-R cognitive architecture, ways in which workload can be predicted based on ACT-R models and the specific model we use in our experiment.

2.1 Code Comprehension

Wyrich defined code comprehension as: *'[. . .] a person's intentional act and degree of accomplishment in inferring the meaning of source code'* [82]. As reading and inferring the meaning of source code is an important part of many software engineering activities, such as debugging or extending existing code, code comprehension is a central topic of software engineering research. The first big effort in researching code comprehension took place in the 1980s and 1990s. In this time, researchers developed multiple models of how a programmer understands code. The most important ones were the bottom-up model by Brooks [10] and the top-down model by Shneiderman et al. [73]. In the top-down model, the developer forms a hypothesis about the code based on beacons, parts of the code that give an indication as to its use, such as identifiers or loop-conditions, and then checks if the code aligns with that hypothesis. In the bottom-up model, the developer reads the code line-by-line and builds up a mental model of the code or simply mentally executes the code line-by-line. These models are still used to this day as the main two modes of code comprehension. Brien et al. showed that programmers do not exclusively use one or the other, but change between the approaches as needed [60].
Starting in the latter half of the 2010s, developments in neuro-imaging techniques like EEG, fMRI, and functional Near-Infrared Spectroscopy (fNIRS) allowed for real-time measurements of brain activity and mental workload. These advances led to new interest in focusing on the internal happenings of the brain during program comprehension [82]. A wide variety of different experimental tasks have been used to examine code comprehension [83]. One common approach is to present participants with a code snippet and ask them to calculate the output of the snippet. In our study, we used this approach in a slightly modified way, as it is a close match to the capabilities of the ACT-R model we use.

2.2 Mental Workload

Mental workload intuitively describes the amount of mental resources used to complete a task. While the general concept of mental workload sees widespread use, there exists no single agreed upon definition of mental workload. In a literature review in 2022 Lungo et al. found as many as 68 different definitions of mental workload [48]. The following are a few examples of definitions of mental workload:

- 'Mental workload refers to the ability of the operator to meet the information processing demands imposed by a task or system' [81]
- 'Mental workload or cognitive load refers to the total amount of human mental effort or memory that is required for the execution of a task' [15]
- 'Mental workload can be defined as the amount of thinking, level of cognitive demand, or thought processing effort required by the worker to meet the physical, temporal and environmental demands of the defined task' [56]
- 'The intensity of mental effort can be considered as an index of mental workload. It may be defined as the total amount of controlled cognitive processing in which a subject is engaged' [62]
- "[...] mental workload can be defined [...] by the ratio of the resources required to the resources available, where time is one of those resources but not the only one" [80]

Despite the lack of a universal definition of mental workload, research in many different fields has investigated mental workload and its impact on human performance [15, 17, 56]. This also includes software engineering research, where many studies investigate the mental workload of software engineers during their work tasks [2, 75], with the main focus often being the mental workload associated with code comprehension [31, 53, 65, 67].

Across areas of research, many different approaches and tasks are used to judge the effects of mental workload on human performance and behaviour. These tasks differ in what sources of mental workload are investigated with them. Some examples of sources of mental workload are working memory load, time pressure [46] or the logical complexity of a task [32]. One task often used in psychological studies to evaluate the impact of mental workload is the n-back task. In the n-back task, participants are presented a stream of stimuli one at a time and are instructed to press a button if the currently shown stimulus matches the stimulus$n$ stimuli before the current one. For $n = 0$ the participant is instructed to look for one specific type of stimuli, for example the letter x and press the button every time that stimuli appears. To differ mental workload, this task is repeated with differing values of n. With increasing n participants have to remember more stimuli. For our study, we use simple snippets to analyse code comprehension. Due to the limitations of the ACT-R model we use (see later in this chapter), increasing workload in these snippets is primarily caused by an increased working memory load (see 3.2.1 for a more detailed breakdown of workload sources). Therefore, we expect some similarities in the workload between n-back tasks and our code comprehension tasks.

As mental workload is a concept internal to the human mind, it is not trivial to measure in a study. The two most common approaches to do so are either to have participants subjectively judge their own workload using questionnaires such as the NASA-TLX [33, 70] or using neuro-imaging techniques to collect data about the internal workings of the mind [14, 65]. Some studies also use other physiological measurements, such as eye-tracking based measures like the amount of refixations on a stimulus, as measures of mental workload [67].

2.3 Neuro-imaging Techniques

Neuro-imaging techniques are physiological measurements that can be used to gain insights into brain activity. The most common measurements are fMRI, fNIRS, and EEG. For an fMRI measurement the participant is placed in an MRI tube, which measures magnetic fields. When a brain region is more active, it is supplied with blood carrying more oxygen. This blood-oxygenation-level-dependent (BOLD) response is detected by fMRI, as blood carrying more oxygen is more magnetic due to a higher iron content [38]. With fMRI, it is possible to pinpoint specific areas of the brain that are more or less active in a given task by comparing the BOLD response to some baseline before the task. However, the BOLD response takes a few seconds to occur after brain activity increases. As such, fMRI has very good spatial resolution and poor temporal resolution [38]. Additionally, fMRI is very expensive and limiting in experimental design as no magnetic objects (such as electric devices) can be placed within the tube and participants must lie still during the measurement [67].

Similar to fMRI, fNIRS measures the increase in oxygenated blood flowing to active brain areas. It uses near-infrared light to measure changes in the oxygen-level of blood in the cortex [47]. As it also measures the BOLD response, fNIRS shares the poor temporal resolution of fMRI. In addition, its restriction to areas near the surface of the scalp worsens the spatial resolution. It does, however, have the advantage that it allows for more movement of the participant, does not restrict the set-up of the experiment as much, and can be used while sitting up [47].

For EEG measurements, electrodes are placed on the participant's scalp to measure oscillating electrical potentials generated by neuron activity in the brain [43]. A change in neuron activity is reflected immediately in the measured potentials, allowing for very good temporal resolution in EEG [43]. Because measurements for EEG only occur on the scalp, it is not possible to clearly find the origin of an electric potential, limiting spatial resolution. Additional advantages of EEG are its comparably cheap usage and the possibility to use it in a normal work environment for software engineers, as it can be used while sitting at a desk and working with a PC. While noise due to electrical impulses generated by muscle movement means EEG imposes some restrictions on movement, they are significantly less strict than those of fMRI and artefacts affect only the data measured at the same time as the movement, with all other data still being usable, even in cases of heavy movement during an artefact.
The activity measured by EEG can be divided into different power bands based on frequency, with different kinds of brain activity causing higher spectral power in different frequency bands. For example, beta band (13-30Hz) spectral power correlates with high stress levels

and anxiety [17]. For mental workload studies, two very important frequency bands are the theta band (4-8 Hz), for which activity in the frontal brain regions correlates with actively working on a problem, and the alpha band (8-13 Hz), for which activity in the parietal brain regions correlates with a relaxed mental state [11, 54]. With increasing mental workload, alpha band spectral power at parietal electrodes decreases, while theta band spectral power at frontal electrodes increases [11]. Therefore, the ratio of frontal theta band power to parietal alpha band power has been used to measure mental workload both in code comprehension studies [54, 66] and in other fields [11, 35]. While other spectral band power based workload measurements exist, we decided to use the ratio of frontal theta to parietal alpha band power (short: theta-to-alpha ratio) in our study due to its prior successful use in code comprehension studies and the heavy emphasis of our study on working memory load, which strongly correlates with theta band power. The combination of these factors makes the theta-to-alpha ratio the most appropriate band power based measurement for our study.

Another option for analysing EEG data are ERPs. For these, we analyse the potentials measured around a specific event, such as presenting a stimulus [49]. In general, it is necessary to aggregate data for the same event over multiple trials to calculate ERPs, because measurements at a single moment in time are strongly affected by noise [49]. The exact number of trials that is recommended to use depends on the investigated component and the expected effect size. Before the aggregation, each trial's data is normalized by using the average over a span of time (often 200 ms) before the onset of the event as a baseline to account for individual differences in brain activity level [49]. ERPs measure the immediate response to an event in the brain, usually stopping after roughly 1 second. When we are interested in the difference between two types of event, we can use ERPs to analyse processes in the brain that occur in one case but not the other. To do so, we create a difference wave, which shows only those parts of the ERP that are present in one case but not the other, thereby revealing differences in the treatment of two stimuli in the brain [49].
When talking about potentials in ERPs it is customary to talk about them as Nx or Px, with N or P indicating a negative or positive potential and x indicating the time in milliseconds since the event occurred. So, an N100 would be a negative potential occurring 100 milliseconds after the event. For specific potentials that have been observed in multiple studies, there often are names that are used specifically for these potentials, even if they do not always occur exactly at that time. ERP research refers to these potentials as components, and a large part of ERP research is focused on what affects these components and how they combine to form the ERP signal. One such component is the P300, which often occurs when presenting a stimulus and is a positive potential that occurs around 250-500 ms after the event. It is believed to be linked to early decision making processes related to the stimuli. Huffmeijer et al. recommend aggregating over a minimum of 60 trials in studies investigating the P300 [36]. The amplitude of the P300 for a stimulus depends on many different factors. As an example, it increases in amplitude for stimuli that differ from the majority of stimuli presented in a study ('oddball-paradigm')[72]. The P300 also increases in amplitude for target stimuli, meaning stimuli that match some prior description, such as a stimuli matching the stimuli n prior in an n-back task. For some tasks, including the n-back task, studies have also found that the P300 amplitude at the Pz electrode shrinks

with increasing workload [14, 72]. In this thesis, we investigate if this relationship holds for a code comprehension setting, i.e. if the P300 does decrease in amplitude as mental workload increases

2.4 ACT-R

Cognitive architectures are models designed to simulate theories of human cognition [46]. In this thesis, we use one such cognitive architecture, ACT-R, to predict mental workload during simple code comprehension tasks. ACT-R was first presented in 1993 by Anderson [4], and has been continuously developed since then. Aside from ACT-R other cognitive architectures exist, such as Soar [45] and Executive Process Interactive Control [55]. In this thesis, we use ACT-R 7.28, the at time of writing newest version of ACT-R. We decided to use ACT-R because there already exists an ACT-R model for a limited subset of Python that is able to predict human performance for a code comprehension task [18], and ACT-R models have been successfully used to predict mental workload in various experiments in other fields, such as simulated Air Traffic Control [46] and memorization tasks [39, 61].

2.4.1 Concept

The basic idea of ACT-R is to simulate human cognition in a model consisting of multiple subsystems, called modules (not to be confused with the programming concept of a module), with each module simulating a specific part of human cognition, and their interactions allowing the simulation of coherent cognition [5]. The model can send requests to a module, which the module then reacts to. Requests are generated either by the model following specified rules, or by the procedural module as part of executing the model. Each module supports a specific set of requests, and has one or more buffers in which information can be stored to be made accessible to the remaining modules. All other information a module has access to is internal information and not accessible to the rest of the model. Each buffer can contain one set of values at a given time. These sets are called chunks and represent the basic unit of information for the communication between modules. Each chunk has an arbitrary number of slots which are name-value pairs, and it is possible to add or remove slots from a chunk [29]. The core ACT-R model consists of eight modules, with the most important ones being the procedural, imaginal, declarative and goal modules [28]. In the following, we explain these eight modules, including the ways requests for them can be generated, how they react to these requests, and what requests they support.

2.4.2 Goal Module

The goal module represents the formulation of goals in the mind and stores an encoding of the current goal of the model (e.g. 'calculate the sum of 3 and 4'). It can also be used to store partial results or other information that is required by calculations, but it is recommended to instead store this information in the imaginal module to better simulate the human mind [29]. This is especially important for the prediction of the execution time of a task, because the goal buffer is modified and accessed instantaneously, while the imaginal buffer is slower

and thus more closely aligns with the reality of human cognition.

2.4.3 Procedural Module and Production Rules

The procedural module takes the central role in the model, as it stores a collection of production rules that can be applied depending on the state of the system [5]. It simulates memory of procedures (e.g. "To turn on the light, flip the light switch" or how to add two numbers). Each production consists of a set of conditions which must be fulfilled for the production to be applied, called the left-hand side, and a set of actions to be taken when the production is applied, called the right-hand side. Possible conditions include a buffer being full or empty, specific slots of the chunk in a buffer having specific values or matching other slots (possibly in a different chunk in another buffer), or a module being available for computation. For each step of the model, the procedural module checks which of the productions are applicable in the current state of the model. If no production is currently applicable, the model pauses until a change in the simulated environment occurs (e.g. another module finishes its work or a new stimulus is presented), checking again which productions are applicable afterwards. If the conditions for exactly one production are fulfilled, that production is executed, whereas if the conditions for multiple productions are fulfilled, the model executes the one with the highest utility. Utility is a value assigned to each production when the model is written. It is possible to enable utility learning, so that the model tries to learn how useful productions are during execution. If utility is equal for two or more productions, the model executes one of them at random. [29]

2.4.4 Declarative Module and Chunk Activation

The declarative module simulates short and long term memory for facts and information (e.g., '3+4=7' or 'The sky is blue') [5]. These facts can be encoded into the system by the modeller, or stored by the model while it is active. The declarative module stores every chunk that is discarded by any buffer in the system, allowing access to previously used information at a later point in processing [29]. To access information stored in declarative memory, a production sends a request to the declarative memory (e.g., An addition result for addends 3 and 4). The declarative memory then puts a chunk that matches this condition into its buffer, making it available for future productions [28]. To decide which chunk to use, if multiple chunks match the given condition, each chunk stored in declarative memory has an activation value consisting of three components: base activation, context activation, and noise. If multiple chunks match the conditions, then the module selects the one with the highest activation [28]. Base activation B_i represents how well the model has learned this chunk through repeated usage of it. It is defined as:

$$B_i = \ln(\sum_{j=1}^{n} t_j^{-d}) \tag{1}$$

Where n is the number of times the chunk was presented. This refers both to the initial creation of a chunk, and each time afterwards a chunk with the exact same slots and values is cleared from a buffer. The time that has expired since the jth presentation of a chunk is represented by t_j. The decay parameter d can be used to modify how fast information decays in the model, and is usually set to 0.5.[29]
Since this approach to calculating base activation is computationally expensive, an approximation can be used instead. This approximation assumes that the presentations of a chunk are distributed roughly uniformly over the time since the chunk was created. The approximation is:

$$B_i = \ln(\frac{n}{1-d}) - d * ln(L_i) \tag{2}$$

where L_i is the time since creation of the chunk, n is the number of presentations and d is the decay parameter [29]. It is also possible to manually set the base activation of a chunk in the declaration of a model. This can be used to simulate long-term memory, by setting the base activation of facts we expect to be in long-term memory (e.g. results of an addition) at a very high value. This ensures that well-known facts can always be accessed [29].
The noise component is defined by a parameter s and is generated by a logistic distribution with mean 0 and variance

$$\sigma^2 = \frac{\pi^2}{3}s^2. \tag{3}$$

Each time a request is made to declarative memory, a noise value is randomly generated from this distribution and added to the base activation of each chunk in declarative memory [29]. It is also possible to add a permanent noise to the activation of a chunk at the time that chunk is generated, but this option is not commonly used in ACT-R models [28].
Context activation can be used to increase or decrease activation based on the current state of the model, such as buffer contents [29]. It is an optional extension of the activation system, and we do not explore the details of it here, since the model used in this thesis does not use context activation [18].
In addition, a retrieval threshold τ can be set. If a chunk's activation is below that threshold, it is not retrieved. This can be used to simulate the model not always recalling every fact after seeing it once. Together with the noise component of activation, this allows us to simulate probabilities of recalling a chunk. Based on the base activation of a chunk B_i, the retrieval threshold τ, and the noise parameter s, the probability of a chunk being recallable can be calculated as:

$$P_{recall} = \frac{1}{1 + e^{\frac{\tau - B_i}{s}}} \tag{4}$$

In addition, the model also simulates that accessing well learned information is faster [29], with the time needed for retrieval being affected by the total activation A_i of the retrieved chunk according to the formula:

$$t_i = F * e^{-f * A_i} \tag{5}$$

where F is the latency factor and f the latency exponent. Both F and f can be used to modify the impact of activation on latency. If no chunk matching the retrieval request has a high enough activation, the time to signal a failure to find is calculated with the retrieval threshold τ as:

$$t_i = F * e^{-f * \tau} \tag{6}$$

In total, there are five parameters affecting the declarative memory, which we can adjust to fit an ACT-R model to empirical behavioural data: the noise parameter s, the decay rate d, the latency factor F, the latency exponent f and the retrieval threshold τ [29].

2.4.5 Imaginal Module

The imaginal module is used to generate new chunks from existing information by specifying the slots to be put into the chunk that is to be generated (e.g., the addends after reading an addition task to store the read information) [29]. The generated chunk is then placed into the imaginal buffer. It is also possible to modify the chunk in the buffer with a production. The main uses of the imaginal module are to store information needed to complete a task, to combine information from multiple sources (e.g. a read word and a heard word that should be compared), and to put information into declarative memory by generating a chunk containing that information and then harvesting that chunk so that it is put into declarative memory [29].

Together, the four previously explained modules simulate the internal processes of the human mind. They can be extended with other modules to allow interactions with a simulated environment. These modules are divided into two categories, perceptual modules allowing the extraction of information from the environment, and motor modules allowing the model to take actions influencing the environment [28].

2.4.6 Perceptual Modules

Perceptual modules are used to perceive stimuli in the virtual environment of the model. They have an additional location buffer, which is used to find the location of a stimulus that the main module can then be directed to attend [28]. Requests to the location buffer can have different conditions, and in response the location of one stimulus fulfilling these conditions is put into the buffer. Then, the module can be given a command to attend the stimuli at the given location by passing the chunk from the location buffer to the module in a production. Perceptual modules also share a behaviour called buffer stuffing, which means that if the model is not busy and the location buffer is empty when new stimuli appear, one of the new stimuli is immediately put into the buffer, allowing the model to detect changes in the virtual environment [29]. Perceptual modules also track which stimuli they have previously attended, enabling them to scan for new or unattended stimuli [29]. The core ACT-R model has two perceptual modules, the visual and the aural module [28].

The visual module simulates human perception. It can look at specific parts of the (simulated) screen, read letters, numbers or words located there and track what objects on the screen it previously looked at [28]. In addition, when moving attention, it is possible to specify the attributes (colour, size, type, content, location) of an object that should be

attended. This can be used to simulate scanning a larger set of items for a specific one (e.g., finding the declaration of a specific variable in code) [29].

The aural module simulates audio perception. It can listen to tones and spoken digits or words [28]. Similar to visual stimuli, sounds can be attended based on their attributes (type, content, onset time, duration). For tones, the model can identify the frequency of the tone, and for words or digits, it identifies the spoken word or digit. It does not automatically map this information to a representation, as that should be done by a separate production after attending a sound (e.g., mapping the digit sound 'three' to the number 3) [28].

2.4.7 Motor Modules

Motor modules are used to interact with the virtual environment of the model. These modules can be given commands as a buffer request, which then causes them to interact with the simulated environment in a way defined by the module [28]. There are two predefined, generic motor modules in the ACT-R architecture: the manual module and the vocal module.

The manual module simulates physical interaction with the environment. It supports usage of a keyboard, mouse or joystick, and can be extended to use different, modeller implemented, devices. It is possible to give individual commands to fingers or hands to move to specific positions, or to use commands that perform specific actions, such as typing a specific letter or clicking a mouse button [28].

The vocal module simulates speech and supports both actual physical speech and subvocalizing (internal speech). The audio module is able to perceive actions made by the vocal module and can identify if a sound it hears was created by the vocal module (including discerning between sub-vocalising and speech) or by the simulated environment [28] .

The core ACT-R model described here can be extended with additional capabilities by adding new modules, allowing for example for more modes of interaction with the environment or for the simulation of a perception of time [28].

2.5 Measuring Mental Workload in ACT-R

The ACT-R cognitive architecture described above allows for the simulation of a wide variety of experimental tasks. However, it does not have a built-in way to predict mental workload during the execution of these tasks. The first attempt at predicting mental workload using ACT-R was made by Lebiere et al. in 2001 [46]. They defined workload as the ratio of time spent on critical unit tasks to the total time spent on the task. In that approach, what constitutes a critical task has to be defined for every ACT-R model individually. One problem with this approach was the limitation that the model either is in a critical task at a certain time, or it is not, with no possible in-between. This makes it impossible to account for

low-intensity, high-duration tasks. To allow accounting for such tasks, Jo et al. constructed a more complicated mathematical model to predict mental workload based on data collected during the execution of an ACT-R model. They built their model on the definition of mental workload as the ratio of resources required for a task to the ratio of resources available [39]. Their approach allows for the calculation of the instantaneous workload $IW(t)$ of the model and individual modules at time t and for the calculation of the total accumulated workload AW of the model and individual modules over the execution of the model. The instantaneous workload of the model at time t is defined as follows:

$$IW_{total}(t) = \sum_i (IW_i(t)) \tag{7}$$

$$IW_i(t) = W_i * E_i(t) * A_i(t) \tag{8}$$

Where $IW_i(t)$ is the instantaneous workload in module i at time t. W_i is the weight assigned to module i, representing different modules contributing differently to mental workload. Jo et al. proposed to use 1 for perceptual and motor modules (W_{pm}), 2 for the goal and procedural module (W_c) and 4 for the declarative and imaginal modules (W_m). The exact values in this set of weights were subjectively decided by Jo et al., but the order of the weights ($W_{pm} \leq W_c \leq W_m$) was based on existing models of cognitive workload. In their validation of the model, they found that different values for the weights showed similar and good correlation with mental workload, so long as the order was kept the same, and $\frac{W_c+W_m}{W_c+W_{pm}}$ was between 2 and 3. The activation function $A_i(t)$ is 0 if the module i is not activated at time t, and 1 if it is. The error weight function $E_i(t)$ is used to increase mental workload when a module faces an error, because perceived workload increases as perceived performance decreases [33]. It returns 2 if the module i is in error at time t, and 1 if it is not. As errors in the specific task, they investigated in their experiment, Jo et al. used failing to retrieve known information and failing to find something in a visual search but for different tasks and models, it is necessary to adjust what exactly counts as an error.

Based on the described instantaneous workload IW, it is then possible to calculate the accumulated workload AW and the average workload W_{avg} as follows:

$$AW_i = \int_0^T IW_i(t)dt = \int_0^T W_i * E_i(t) * A_i(t)dt \tag{9}$$

$$AW_{total} = \sum_i AW_i \tag{10}$$

$$W_{avg} = \frac{AW_{total}}{T} = \frac{\sum_i \int_0^T W_i * E_i(t) * A_i(t)dt}{T} \tag{11}$$

Jo et al. used the average workload in ACT-R to predict average mental workload for human participants, measured using the NASA-TLX questionnaire, and found a strong linear correlation between predicted and measured mental workload using simple linear regression.

Building on this approach, Park et al. proposed formulas to predict the six sub scales of the NASA-TLX [63]. In their approach, the activation of different subsets of modules is used for each sub scale. For example, the physical demand is predicted based only on workload in the motor modules. Another extension of the workload prediction model was

created by Oh et al. [61] who added a decay function to the model to allow the calculation of the instantaneous workload of the model at a certain point in time, while considering workload that has accrued in previous tasks. Crucially, they did not use this model to predict workload directly, but instead used it to inform the behaviour of the ACT-R model, specifically in deciding when to switch between tasks.

2.6 ACT-R Model of Code Comprehension

In this thesis we use an ACT-R model that was developed by Closheim [18] to simulate code comprehension in simple Python snippets. In the following, we explain how this model operates, what it is capable of and what limitations it has.

2.6.1 Capabilities and Limitations

The model simulates a line-by-line reading of an input code snippet, pressing the space bar after each line to proceed to the next. This continues until the model reads a print statement, at which point it uses its virtual keyboard to type the result of the print statement, confirms with the space bar, and finishes the trial. The model only supports assigning values to variables, simple arithmetical operations (addition, subtraction, multiplication, and division of integers, with operands and results in the range of minus 20 to plus 20) and print statements. It allows the use of one or more variables in an arithmetical operation, and to assign the value of an arithmetical operation to a variable. The model only supports a single variable or integer as a parameter in print statements. Thus, it is not possible to directly print the value of an arithmetical operation; instead, we have to store the result in a variable and then print that variable. In addition, the model does support combining multiple arithmetical operations in a single line, but does not support priority rules for arithmetical operations, instead executing operations from left-to-right. The model has five hyper-parameters that can be used to fine-tune it. The available hyper-parameters are the already discussed retrieval threshold rt, decay rate d, noise parameter s, latency factor F, and latency exponent f [18].

2.6.2 Internal Mechanics

The model initially contains each arithmetic operation it supports as a chunk in declarative memory. This was done instead of implementing productions to execute operations, because the focus of the model is supposed to be code comprehension, not mental arithmetic speed [18]. The model simulates the reading of lines of code by scanning them from left to right, reading each variable name, number, and operator as a word. When the model encounters a variable declaration, it uses the imaginal module to create a new chunk storing the value of the variable and its name. This chunk is then immediately harvested and put into declarative

memory. For this behaviour, the model does not check if there already is a chunk with the same variable in declarative memory. This means that, in such a situation, chunks with different values and the same name can exist in declarative memory, with activation deciding which of them will be retrieved by any future recalls of that variable. If the model encounters an arithmetic operation, it calculates partial results as soon as possible. If it encounters a variable, it retrieves the value of that variable from declarative memory, and if retrieval is successful, it uses the retrieved value in the calculation. After reading an operation (value-operator-value) the model retrieves the matching chunk from declarative memory, and if retrieval is successful, it uses the result value of the retrieved chunk as the result of the operation. If the retrieval of a variable or operation fails, the model ceases to operate, as without the retrieved value present no production rule can be applied to the current state. If the model reads a print statement, it evaluates the expression passed to the statement and then types the value of that expression, confirming afterwards by pressing the space bar. As can be seen in this description, the model currently only supports bottom-up comprehension because it reads and mentally executes code line-by-line.

2.6.3 Snippets

The snippets used by Closheim were divided into four categories, with each category representing a different source of difficulty in understanding code snippets or the interaction between two sources. Due to the model's limitations, only some sources of complexity could be investigated. Specifically, Closheim chose repeated assignment of values to a variable and the distance of a variable to its usage as the sources of confusion. Notably, given that only variable assignments were used as code, the distance of a variable can be expected to align well with working memory load during usage, since participants do not know which variables will be used later in the snippet, and thus have to keep each declared variable in working memory. This shows some conceptual similarity with an n-back task, which also requires the participant to keep previously shown stimuli in working memory, with the number of stimuli increasing with increasing task difficulty. Overall, Closheim found that, after fine-tuning the model, it simulated answering time data for participants with a high accuracy, but had issues in matching the error rate of participants. In addition, when the model made errors, they often differed from the ones participants made. This was the case because the model only has two ways to produce a wrong result: It fails to recall a chunk from memory, in which case it outputs nothing, or it recalls an already overwritten value for a variable, in which case it calculates the result with this old value. In practice, all participants answered something, possibly guessing a result when failing to remember a value. Additionally, some of the wrong answers by human participants may be due to errors in calculations, which cannot happen in the model. They also found that, for the snippets they used, the effects of the sources of complexity were not statistically significant in most cases.

3

Methodology

In this chapter, we introduce our research questions and the experiment we conducted to answer them.

3.1 Research Questions

Closheim found that ACT-R can simulate answering time and, with some limitations, answering correctness for program comprehension tasks. We conducted a non-exact replication of their study, using a new set of code snippets. We compared the correctness and answering time of the ACT-R model with data from human participants and evaluated how well the ACT-R simulation matches human data. This was necessary to prepare our planned investigation of the predicted workload, because Closheim's model has only been verified on a limited set of snippets and participants in an online setting. If the model does not generalise well to the snippets we used or our experiment setting, we need to retune the hyper-parameters of the model. We required an ACT-R model that simulates human data as well as possible since our workload prediction is based on an ACT-R model simulating human behaviour. We investigated if Closheim's ACT-R model of code comprehension accurately simulates code comprehension on the snippets we used in our study, and fine-tuned the parameters of the ACT-R model to improve how well it simulates human data. To this end we posed the following research question:

RQ0: How accurately does the ACT-R model predict answer correctness and answering time in our snippets?

To evaluate the viability of using the ACT-R model to predict mental workload during code comprehension, we examined if it can be used to predict the average subjective workload of human participants for our snippets. To do so, we calculated the correlation between the average subjective workload of a snippet and the average predicted workload for that snippet. Additionally, we examined what the specific relationship between predicted workload and subjective workload is. We checked if the linear relationship between subjective workload and predicted workload, found by previous studies using the workload prediction algorithm, holds in a program comprehension setting. We posed the following research question:

RQ1: How well does the workload predicted based on the ACT-R model correlate with subjective workload reported by human participants?

As previously discussed, studies found that, for a variety of tasks, the P300 component at the Pz electrode in response to a stimulus reduced in amplitude with increasing workload.

Notably, the program comprehension task we used shows considerable similarities to some of these tasks, such as the n-back task. We examined if there is a relation between mental workload and the average P300 amplitude caused by a new line being shown in the program comprehension task. To increase the accuracy of our ERP data, we aggregated snippets into easy, medium, and hard groups based on participants subjective workload answers, and compared the amplitude of the average P300 response for the last line of snippets in these groups. We chose the last line because we expected to be able to see the effect of the workload of the entire rest of the snippet in the response to this line. Therefore, we expected to see the differences between snippets most clearly in this line. We also analysed the P300 for the first line in each group, where we expected the values to be equal for all groups. We did this because the approach of measuring workload using the P300 amplitude is new for program comprehension tasks, and we wanted to ensure that it not only is able to differ between workloads, but also give similar results for equal workloads. With this analysis, we intended to answer the following research question:

RQ2: How does the P300 caused by reading a new line differ in amplitude with increasing mental workload?

So far, ACT-R-based workload predictions have only been used to predict subjective workload measurements. To inform further development of an ACT-R based workload prediction in code comprehension, it is important to know if the predicted workload also correlates with neuro-imaging measures of mental workload. The existence or absence of such a correlation could inform us of potential aspects of workload that the ACT-R prediction does currently reflect well or badly. If we find EEG-based measurements that the model does not predict well, this could inform the future development of the ACT-R model, based on what aspects of mental workload that measurement aligns well with. We examined the correlation between the predicted workload and two EEG-based measurements of mental workload, namely the theta-to-alpha ratio and the average P300 amplitude. In this way, we covered both frequency based methods and ERP based methods, two of the primary categories of EEG analysis methods. With this approach, we investigated the research question:

RQ3: How well does the workload predicted based on the ACT-R model correlate with EEG-based mental workload measurements?

One big advantage of EEG data is its high temporal resolution. However, ACT-R studies that did use EEG in the past compared data over longer time frames, and focused on matching activity in specific brain areas to activity in ACT-R modules. Therefore, these studies did not benefit from the temporal resolution of EEG. We made use of the high temporal resolution of EEG data to examine the predictability of mental workload for individual lines using the ACT-R-based workload prediction. This could extend the possible future uses of an ACT-R model of program comprehension. It would be of great benefit to be able to identify individual lines that cause high mental workload during comprehension and mark them for a developer to examine and possibly rewrite. For this, we calculated the theta-to-alpha ratio for individual lines of code and compared them to the ACT-R based workload measurement, by limiting the formula to only the time in which the snippet is shown to the model. Additionally, we investigated the relation of both our workload measurements and the ACT-R prediction to different sources of mental workload. As this required an analysis of

the data in aggregate over multiple trials, we also can include the P300 amplitude in the ERP in this part of the analysis. For the analysis of individual lines, we did not have enough participants to get reliable ERPs in the aggregate, and therefore omit the P300 amplitude from that analysis.

RQ4: Are there significant differences in workload measured with EEG-based measurements between lines? If so, can the workload prediction based on EEG predict these differences?

3.2 Operationalization

In the following section, we describe the setup for our experiment and the tools we use to evaluate the data. Because we evaluate the match of the ACT-R model's simulated data on real data, we conduct this experiment with human participants and simulate it with ACT-R. Unless further specified in the following, 'participant' refers both to a human participant and the ACT-R model simulating the task.

3.2.1 Snippets

```
k = 3
x = 2
c = k * x
print(c)
```

```
u = 7
s = 8
q = u - 3
u = s / q
s = q / 2 * 3 + 5 + u
print(s)
```

```
k = 4
c = 9
c = 3
p = 6
k = 7
p = 5
m = k + c
k = p - c
k = m * k
print(k)
```

Figure 3.1: Some examples of snippets from our study. On the left is an easy snippet, in the middle is a hard snippet with calculation length as the primary source of mental workload and on the right is a hard snippet with working memory load as the primary source of mental workload

We designed our code snippets to cover a wide span of induced mental workloads, because this allows us to evaluate the ACT-R model of code comprehension on a wide span of possible workloads. In creating these snippets, we had to limit ourselves to the capabilities of the ACT-R model created by Closheim, which strongly limits the possible sources of mental workload in a snippet. As previously explained, the model is only able to process variable declarations, redeclarations, and calculations (addition, subtraction, multiplication, or division) in the range of -20 to 20. As such, we based the basic layout of our snippets on those used by Closheim [18] and on snippets used in an ACT-R study by Dörrzapf [23]. We also directly adopted eight snippets from these studies for our study

(either the treatment or the control version of each of their snippets). Each snippet consists of one or more variable declarations and redeclarations, using both integers and calculations. The calculations use previously declared variables and integers as operands. The last line of each snippet is a print statement, printing the value of exactly one variable. In these limited snippets, the only relevant sources of mental workload that can occur are:

- The number of variables declared, because the participants need to keep the values for every variable in working memory.
- How often variables are redeclared, because every redeclaration requires the participant to remember the new value, and not confuse it with the old value of a variable.
- The difficulty of individual calculations in the snippet, since calculations are limited to the range of -20 to $+20$ we expect that there is only a slight effect between individual calculations, but extending a calculation within one line most likely would increase mental load.
- The amount of calculations in a snippet because a line with a calculation likely requires more mental load than one with a simple assignment. This is the case both due to the mental load associated with the calculation itself and the mental load of accessing variables used in the calculation.

In creating our snippets, we used these four causes to affect mental workload for a snippet. We varied all of these factors considerably between snippets, with snippets featuring between 2 and 7 variables with redeclarations making up between 0% and 72.73% of non-print lines. Calculations were featured in between 20% and 83.33% of non-print lines, and the maximum length of calculations varied between 2 and 5 operands. The varying length of calculations is important as it is the only cause of mental workload that is largely independent of snippet length, since increasing any of the other factors requires increasing the line count of a snippet. Such a measure existing is important because it ensures our results can apply to workload in general, and are not just caused by an increasing snippet length. The length and difficulty of calculations is also the only source of mental workload that does not primarily affect working memory load. Due to the model not including order of operation rules in its simulation, all calculations were chosen in such a way that a simple left-to-right reading matched the correct order of operations (e.g. a line first contains all multiplication and division, and then all addition and subtraction).

To ensure snippet workload was unaffected by variable identifiers as much as possible, we created three sets of single letter identifiers. For each of these sets, we checked that no easy to remember groups of letters (such as a, b, c or x, y, z) and no easily confused letters (such as p, q or i, j) were present. For each snippet, we then randomly assigned one of these sets, and replaced each variable identifier with one of the identifiers in the set. The specific identifier for a given variable was also chosen at random from the set, rather than in a specific order, to prevent participants learning the order of identifiers in a group over time. This replacing of identifiers was applied to all snippets, including those from the studies by Closheim and Dörrzapf.

To ensure the suitability of the snippets for evaluating mental workload, we conducted a

pilot study with two participants and collected subjective workload data to evaluate the perceived difficulty of these snippets. For measuring subjective workload, we used three of the six sub-questions of the NASA-TLX (Mental Demand, Performance, and Frustration). We decided to only use these three questions because the other three are of limited relevance to our experiment. Our tasks require almost no physical movement (only typing numbers and pressing a button), we set no time limit that could cause temporal demand and with only one type of demand the effort scale would reflect only that demand. We also did not use the ranking questions of the NASA-TLX, as those would significantly increase the time it takes to perform the workload evaluation after each snippet and the raw NASA-TLX score typically correlates strongly with the weighted one [12]. Thus omitting the ranking questions saves a lot of time in the experiment without significantly affecting our results.

In our pilot study, we found that, for all three NASA-TLX sub-scales, a wide span of workload was covered. The subjective workload, as measured with the NASA-TLX, differed between a maximum of 89.83 and a minimum of 4.5, with a diverse distribution of workload over all snippets covering the span between the two extrema. This indicated to us that the snippets we chose covered a sufficiently large span of induced mental workloads, to allow us to proceed with our study without major alterations to the induced workload of our snippets. This was further confirmed by the time and correctness data, with snippets differing significantly in the time needed to comprehend them, and with both snippets with two correct answers and those without any correct answers by pilot participants existing. Based on these findings, we decided to proceed with our study using the same snippets we used in the pilot study. When comparing the three sub-scales of the NASA-TLX we employed, we found that while there was some correlation between them, they also showed considerable differences. For example, one of the pilot participants was very confident in their performance and therefore rated many snippets with a 90+ performance rating, while giving considerably more diverse answers in the other two scales. As such we decided against further simplifying the NASA-TLX for our study and used all three scales.

3.2.2 Workload Prediction

To predict mental workload during code comprehension, we used the workload prediction algorithm by Jo et al. [39]. We decided to use their algorithm, without the changes introduced by Oh et al. [61] and Park et al. [63], because these extensions do not meaningfully apply to our tasks. Since only a few sub scales of the NASA-TLX are relevant for code comprehension, the sub-scale prediction introduced by Park et al. would not be helpful, and because we allow our participants to pause between each snippet to reset their workload, the decay function introduced by Oh et al. to allow including workload from previous tasks cannot be applied to our study. Therefore, we applied the algorithm by Jo et al. to predict the average workload on the ACT-R model of code comprehension by Closheim [18]. Before applying this algorithm, there were some choices we had to make for parts of the algorithm. We needed to decide what weights to use for each of the modules of the ACT-R model, and what behaviour in each module we count as an error. For the weights, we decided to use the default weights proposed by Jo et al., since those performed well in their validation study and there is no immediate reason to believe a different set of weights would allow for a better prediction.

Regarding the errors, we decided to count failed information retrieval (e.g. the model fails to retrieve the value associated to a variable from declarative memory) as an error, but not failed visual search (e.g. a retrieval failure in the visual-location buffer), since that is used in the model to detect a line end, which is not something that would be perceived as an error by a participant. A failed retrieval was chosen to count as an error for the workload prediction, because it is a type of error that is clearly detectable for the model, and would also be noticeable for a human encountering the same error. By contrast, while retrieving a wrong value for a variable is still a possible cause of a wrong answer, it is not an error that is immediately noticeable to a human participant, and thus is not counted as an error for the workload prediction. It is important to note that, while the model currently does not simulate many sources for errors that humans may make, most of the not simulated error sources are not necessarily ones that would count as errors in the workload prediction algorithm, as ,for example, a miscalculation is not immediately noticeable to a human that just made it, and the error function is primarily intended to model the impact that noticing errors has on human performance and workload estimation.

3.2.3 Participants

We recruited participants at Saarland University by advertising with posters and in a lecture. Each participant received 15 Euros as compensation for their participation. As a prerequisite for participating in the study, all participants were required to have basic programming knowledge. Participants had to be right-handed and not have any known psychological or neurological conditions. This was necessary because brain activity, and therefore also EEG data, is affected strongly both by a participant's handedness and by many psychological and neurological conditions. Brain activity is also affected by many drugs. Therefore, participants that had taken drugs in the 24 hours before the study were excluded from participating. Participants were informed of these conditions when signing up for the study, and again before the study started. In total, we invited 34 participants (23 male, 11 female). All of our participants were students (6 Bachelor students, 26 Master students, 2 PhD students), with 7 of them reporting working a computer science related job in addition to their studies. The average age of our participants was 24.56 with a standard deviation of 2.98. All participants judged their overall programming experience as medium to good (3 or 4 on a 5-point Likert scale). We failed to collect EEG data for two participants due to technical difficulties: for one participant, no data was recorded, while for the second participant, the EEG lost connection to the recording software during the experiment. Due to the missing EEG data we omitted these two participants from all analysis regarding EEG data. We did include their data for analysing behavioural data and subjective workload data, as there is no reason to believe that this data was affected by the issues with the EEG data.

3.2.4 EEG

We used an Unicorn Hybrid Black 8-channel headset[1] to record the EEG data. The electrodes of this headset are placed according to the international 10-20 system at the locations Fz, C3, Cz, C4, PO7, Pz, PO8 and Oz. The headset can be used both as a wet and a dry EEG. We decided to use it as a wet EEG, since this allows for more accurate data collection, while the increased set-up time for 8 channels is manageable. Data was collected at a sampling frequency of 250 Hz, using the Unicorn Recorder software. For the later processing of the data we used the Python library MNE [50]. After collection, the EEG data was filtered with a notch filter at 50 Hz to remove power line noise. Additionally, we applied a bandpass filter between 1 and 30 Hz to remove low-frequency drifts and high-frequency noise [68].

3.2.4.1 Artefact Rejection

After filtering, we inspected the data and manually marked periods that were affected by artefacts. We identified artefacts based on their effects on the EEG recording, such as sudden shifts in voltage, short bursts of high frequency noise, or activity with a far higher amplitude than the rest of that participant's data. For each artefact we marked the time frame in which it affected the EEG data and which channels it affected. If a single channel was affected by artefacts throughout a recording, we marked that channel as bad, as it was likely not properly connected with the participant's scalp. Afterwards, we interpolated the EEG data of channels marked as bad, using spherical spline interpolation [69]. The same was done for those channels affected by artefacts temporarily, which we interpolated only during the timespan of the artefact. This allowed us to proceed with our analysis, even if some of our data was affected by artefacts. Importantly, we did not interpolate whole channels if two or more channels were bad, because with only 8 channels total, the impact of interpolated data on our total data would be too large. Instead, participants for whom more than one channel provided bad data were excluded from future analysis. This excluded 3 out of 32 participants. For the artefacts we found, the affected channels were interpolated in the relevant section unless that would cause three or more channels (including those interpolated as bad channels) to be interpolated at the same time. Our approach on the remaining data differed for the two EEG measurements we used, since ERPs and band power analysis can deal with different types of artefacts differently well.

3.2.4.2 ERP

For our ERP analysis, we applied further artefact rejection, as the averaging process of ERP data is very susceptible to being affected by a few outliers. First, we applied common average referencing [1] to remove artefacts that affected all electrodes, while keeping the data relevant to the ERP response. For each line, we then investigated specifically the data relevant for the analysis of the P300: that at the Pz electrode and between 200 ms before

1 https://www.gtec.at/product/unicorn-hybrid-black/

stimulus onset and 1000 ms after stimulus onset. As the stimulus onset, we chose the moment the line appeared on screen. We discarded any trials that included any change in what was presented on screen other than the stimuli line being presented. This meant any trials that featured the stimuli line being presented for less than 1000 ms were excluded. In addition, any trials in which the baseline duration included the presentation of anything except the snippet introduction (for line 1) or the directly preceding line (for any line after line 1) were excluded. These exclusions were necessary because a change in the content presented on the screen would likely mean that brain activity not related to the stimuli line would be included in the ERP. Before the artefact removal each category contained 870 trials (30 snippets and 29 participants). This removed 0 trials for the first lines of snippets, 5 trials for the last lines of snippets, and 463 trials for the print statements. Given the high amount of trials discarded for the print statements (53.22% of all trials), we decided not to proceed with our planned analysis of the ERP for these lines. Afterwards, we discarded any trials that included an artefact that we marked as affecting three or more channels in this time span, or that included a marked bad channel and a marked artefact affecting at least two channels. This excluded 42 trials for the first lines, and 38 trials for the last lines. These were excluded because the data distorted by artefacts would heavily distort the grand average ERP waves. Additionally, any trial in which the Pz electrode was interpolated as an artefact or bad channel was discarded, because, as we only analyse the ERP at Pz, including these trials would mean analysing fully interpolated data. In this step, we discarded 108 trials for the first line of a snippet and 110 trials for the last line of a snippet. Finally, all remaining trials were checked for a number of conditions, and all trials that violated any of these conditions were excluded. All data points in a trial had to be below an amplitude of 70 μV, the difference between two consecutive data points was not allowed to exceed 30 μV, and the peak-to-peak amplitude in a sliding window of 100 data points with a step size of 25 had to be below 100 μV. These steps were necessary to ensure we remove even those artefacts that our manual analysis might have missed, because ERP measurements can easily be influenced by even just a few artefacts being included in the data. This excluded 84 trials for the first line of a snippet and 52 trials for the last line of a snippet. In total, we excluded 234 (26.90 %) trials for the first line of a snippet and 205 (23.56 %) trials for the last line of a snippet. As already explained, we decided not to conduct an originally planned analysis of the lines with print statements. In total, 65.06 % of those trials would have been excluded.

For the remaining trials, ERPs were calculated around the stimulus onset of a line being presented. Each epoch was computed relative to a baseline of the average of the 200 ms before stimulus onset, and reached from 200 ms before stimulus onset to 1000 ms after stimulus onset. To obtain a measure of the P300 response for a group of lines, we first computed an ERP wave for each of the lines, representing the average ERP response for this line over all participants. Then, we took the average P300 amplitude of this ERP wave by computing the average value of the wave between 300 and 500 ms after stimulus onset. We then calculated grand average ERPs for a group as the average over all ERPs in this group. When comparing the means of the P300 amplitude across multiple groups of snippets, we used the Kruskal-Wallis test, because the P300 amplitude was not normally distributed for all groups. If we found a significant difference in the means of the groups, we then used the Mann-Whitney-U test to compare pairs of groups.

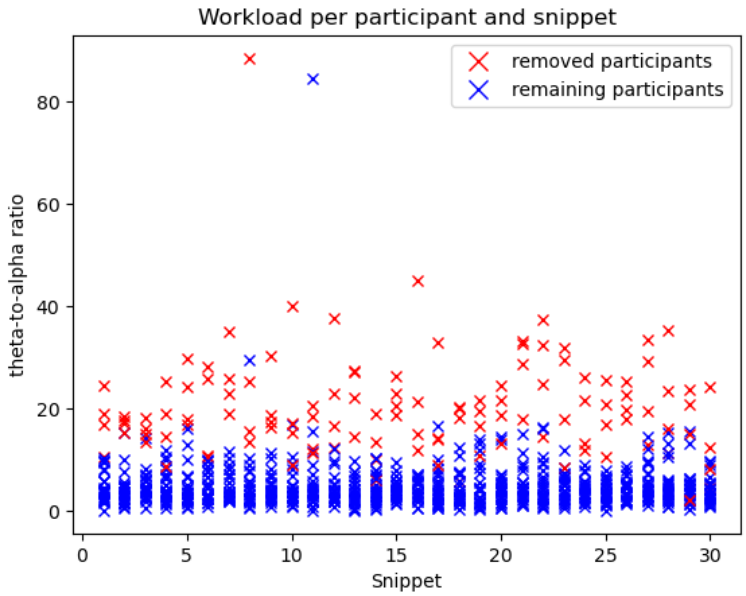

Figure 3.2: Scatter plot of the theta-to-alpha ratio per snippet for each participant. One marker represents a single trial for a single participant. The red markers are the trials for those participants who we omitted as outliers, due to half or more of their values being outside 1.5 times the interquartile range.

3.2.4.3 Theta-to-Alpha ratio

To obtain the theta-to-alpha ratio for a snippet, we first calculated the frontal theta (4-8 Hz) band power (measured at Fz) and the parietal alpha (8-13 Hz) band power (measured at Pz) using the Welch method [41]. The band power for a snippet was calculated over the entire duration of its presentation, from the onset of the first line of the snippet to the confirmation of the answer after the print statement. To obtain a workload measurement, we then calculated the ratio of these two values. We only excluded participants with two or more bad channels from our analysis, as band power analysis is generally more robust against artefacts than ERPs, because it is calculated over a longer time frame instead of averaging values at the scale of milliseconds. However, while less susceptible to outliers in the original data, it can happen that the band power analysis produces results that are clearly outliers from the remaining data. This can be caused by more complex and frequent artefact patterns, or by actual differences in brain activity between participants. In either case, for an analysis aggregating the results of multiple participants, it is necessary to remove these outliers. To find such outliers, we investigated the theta-to-alpha ratios for each participant. We removed all participants for whom the theta-to-alpha ratio for half or more of the snippets were outside 1.5 times the inter-quartile range, as in such a case the remaining data is likely also affected by the cause of the outliers. Using this method, we found four participants for whom their workload data were clear outliers compared to all other participants, and who we therefore excluded from further analysis regarding

the theta-to-alpha ratio (see Figure 3.2). Afterwards, we removed, for each snippet, all participants whose data was outside 1.5 times the inter-quartile range of the remaining data for that snippet. This removed a total of 40 samples or 5.33% of the remaining data. The highest amount of samples removed for a single snippet was 4, leaving us with at least 19 samples for each snippet in our evaluation.

3.2.5 Model Evaluation

For most of our research questions, we need to evaluate how well the ACT-R model fits the empirical data. In this section, we outline how we evaluate this aspect for both behavioural data and mental workload measurements.

3.2.5.1 Behavioural Data

To evaluate how well the model's simulated answering time and correctness match onto empirical human answering time and correctness, we required metrics that can give us a value for the similarity of two distributions. We used the same metrics used by Closheim to ensure comparability between our results. For comparing the similarity of the time distributions, Closheim used the Kolmogorov-Smirnov test [40]. This test was appropriate for this analysis because we wanted to evaluate the similarity of the entire distribution, not just the mean of the distribution, and the time data is continuous. If the ACT-R model accurately simulates human cognition in program comprehension tasks, the distribution of ACT-R results should match the distribution of human results. For evaluating the similarity in the error rate, Closheim used the Chi-Squared test because the only way to evaluate the error rate is as a singular value for each snippet. Comparing the types of errors was not possible, because the model is not capable of replicating most types of errors. It only produces errors caused by wrong or failed retrievals.

3.2.5.2 Workload Prediction

For evaluating how well the ACT-R model predicts a workload measure (subjective workload, alpha-to-theta ratio or P300 amplitude), we first calculated the correlation of the average predicted workload and the average of the workload measure over all participants for a snippet (RQ 1 and 3) or line (RQ 4). For these correlations, we used Kendall's tau, because it can deal with data that is not normally distributed and can handle duplicates better than other measures of correlation. In cases where we found a significant correlation, we then conducted a linear regression to analyse if a linear relationship between the workload measure and the ACT-R prediction exists. We theorised the existence of such relationships based on the findings by Jo et al., who found a linear relationship between subjective workload and the ACT-R-based workload prediction for some experimental stasks [39]. The linear regression gives us values a and b, so that

$$TLX_{pred} = a * W_{avg} + b \tag{3.1}$$

for all snippets. To evaluate the fit of the linear regressions to our data, we report the Root Mean Square Error and R^2-value for each regression. The R^2 value is important for

comparisons between the regressions because the different workload measures use differing scales. If we found that the linear relationship does not hold for a measure, we investigated other possible relationships between the workload measure and the predicted workload.

3.2.6 Experiment

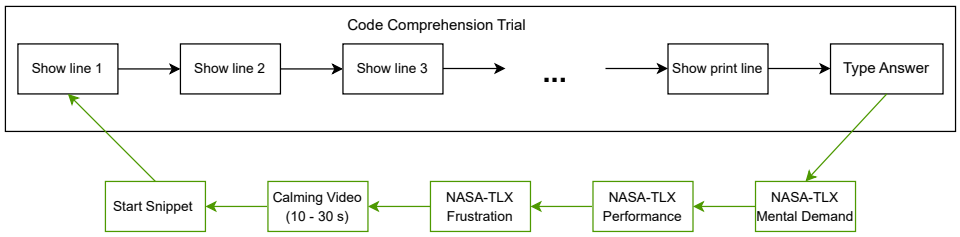

Figure 3.3: The way in which a single trial of our experiment was presented to the participant. The parts marked in green were only presented to human participants. In total each participant was presented 30 such trials.

Each participant was presented with all 30 of our snippets in a random order, which followed some constraints. We randomized our data to minimize the impact of learning and fatigue effects on our data. The constraints were focused on preventing excessive clustering of easy and hard snippets as the participants of our pilot study reported that multiple hard snippets directly after each other led to frustration and task disengagement, while multiple easy snippets led to boredom and task disengagement. Based on the subjective workload feedback from our pilot participants, we identified 5 easy and 10 hard snippets, for which we enforced that:

- No group of 4 consecutive snippets in an order contains more than 2 hard snippets
- No group of 3 consecutive snippets in an order contains more than 1 easy snippets
- Each third of an order contains no more than 4 hard snippets or more than 2 easy snippets.

The ACT-R model was shown the snippets in a constant order, as it was reset between each snippet, and therefore could not be affected by learning or fatigue effects. Figure 3.3 shows how a single trial of our experiment was conducted. Snippets were presented line-by-line, with only one line at a time being displayed. For human participants, we used the presentation software PsychoPy[2] (v.2023.1.3) to present the experiment and record behavioural data, while for the ACT-R model, we used the built-in simulated environment (screen and keyboard) to present the experiment. The line-by-line presentation was necessary to ensure the human participants used strict bottom-up comprehension, as the ACT-R model is also only capable of bottom-up comprehension. Additionally, since the model is unable to go back to a previously read line the human participants also were not given the option of doing so. To proceed to the next line, the participant needed to press the enter button. After reaching the print statement of a snippet, participants hat to press enter again. Then,

2 https://www.psychopy.org/index.html

they were prompted to type the printed value and press enter to confirm their answer and finish the presentation of the snippet. Human participants were instructed to use the numpad and the enter button on the numpad for their inputs. This was done to ensure minimal movement during the experiment, to limit noise in the EEG data. To better match the ACT-R model to the empirical data, we also made a minor modification to Closheim's model by changing the relevant buttons from space-bar and the number keys on top of the keyboard to the enter and number keys on the numpad. Before the first line of a snippet was presented, human participants were shown a screen counting the number of snippets presented so far. Only once the participant pressed enter did the first line of a snippet appear. This was done because, in our pilot study, participants reported that it was frustrating not to know how far along in the study they were. In addition, this ensured that participants were ready when the presentation of a new snippet started, and ensured a neutral baseline for ERP recordings of the first line. After answering a snippet, a human participant was asked to rate their mental demand, performance, and frustration during the task on scales of 1 to 100. Each of these questions was asked separately, using a slider that could be used to enter any value between 1 and 100 with a mouse. As the EEG recordings in these time slots were not used in our analysis, the additional noise this movement causes was not problematic. After the NASA-TLX questions, human participants were shown short videos of fish swimming in a fish tank to reset their mental workload. We decided to use fish videos because watching fish has been shown to have a calming effect [25]. These videos were shown for a minimum of 10 seconds, after which the participants could choose to continue by pressing enter. We showed a small indicator on the screen, so that participants knew that they could proceed if they wished to. If they did not continue on their own, the experiment continued automatically to the introduction screen of the next snippet after 30 seconds. As the ACT-R model was reset after each snippet, and the workload prediction is not affected by prior tasks, no specific action to reset its workload was necessary.

In total the experiment took roughly an hour and 15 minutes for a human participant, although it is notable that a large variance in that duration exists. This variance had two primary causes: First, the set-up duration necessary to ensure proper connectivity between the electrodes and the scalp varied between participants. Secondly, the variance in answering time for the tasks varied strongly between participants. The experiment set-up took roughly between 15 and 45 minutes while participants needed between 30 minutes and 1 hour for the actual experiment.

4

Evaluation

4.1 Results

In this section, we present the results of our study. In our statistical analysis we applied a significance level of $\alpha = 0.05$. To account for the effect of multiple tests all reported p-values were corrected using a False discovery rate (FDR) correction using the Benjamini-Hochberg method [8]. When analysing correlations with Kendall's tau, we used the correlation levels of weak (0.1 to 0.3), medium (0.3 to 0.5), strong (0.5 to 0.7) and very strong (0.7 to 1.0) for both positive and negative correlations.

4.1.1 Research Question 0

To evaluate how well the ACT-R model by Closheim simulates programmers during program comprehension, we compared the answering time and correctness for each snippet between the model and programmers. Overall, we found that the correctness of the model by Closheim differed significantly from programmers' correctness (see Figure 4.1 (bottom left)), heavily underestimating error rate on some snippets and overestimating it on others (mean absolute error: 29.01%, standard deviation: 22.52%). Notably, many of the wrong answers the model produced were caused by it failing a recall and stopping to process the snippet, as it cannot proceed without a value for a variable. In total, missing answers were the cause of 418 of the 537 ACT-R runs that yielded a wrong result. By contrast, almost all errors by programmers were wrong answers instead of missing answers, with only 3 out of 376 wrong answers by humans being non-answers. When investigating the time needed by ACT-R compared to programmers, we found that the ACT-R model significantly underestimated the time required for a task (mean absolute error: 21.519s, standard deviation: 13.346s), as can be seen in Figure 4.1 (top left). Additionally, when investigating the distribution of time for each snippet, we found that, for most snippets, the ACT-R time was distributed much tighter than human participants (see Figure 4.2). Overall, we found that the model by Closheim simulates both answering time and correctness with significant errors for the snippets we used. Because the ACT-R based workload prediction is based on the assumption that the ACT-R model used is a good approximation of human performance in a task, we decided to fine-tune the model on the behavioural data of our study.

We fine-tuned the activation noise parameter, retrieval threshold, and latency factor, as that showed better results in the work by Closheim than optimizing only the activation noise parameter. We decided to only optimize the model regarding the time needed for snippets,

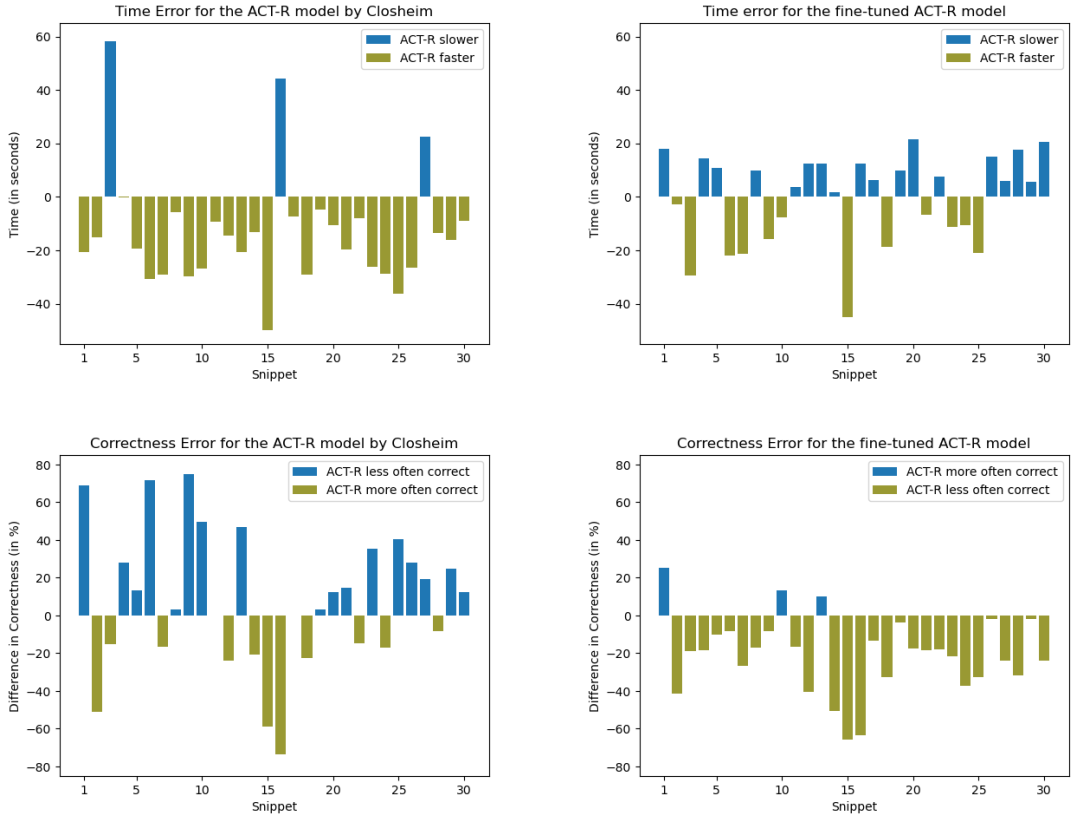

Figure 4.1: Difference in the answering time and correctness for the ACT-R models compared to empirical data. On the left is the data for the ACT-R model by Closheim and on the right is the data for the fine-tuned model we used.

as attempts to optimize for both time and correctness did not converge in Closheim's work. For our optimization we used the same approach as Closheim, a Bayesian-Optimizer with 50 starting points and 1000 iterations with the squared sum of the Kolmogorov-Smirnov statistic over all snippets as the optimization target. The best iteration had a square sum Kolmogorov-Smirnov statistic of 6.89 and first occurred at iteration 285. This iteration had values of 0.581 for the activation noise parameter, -2.444 for the retrieval threshold and 2.873 for the latency factor.

This fine-tuned model shows a significantly better fit on time with a lowered Kolmogorov-Smirnov statistic and lower average difference in time between ACT-R and programmers (mean absolute error: 13.920s, standard deviation: 8.774s) which can be seen in Figure 4.1 (top right). In addition, the distributions for individual snippets also match significantly better (see Figure 4.2). Compared to the model before fine-tuning the time distributions are more spread out and fit programmers better, as indicated by the lowered Kolmogorov-Smirnov statistic.
Even though we did not optimize this model for matching programmers' error rate, we found that it matches the error rate better than the model by Closheim, with a Chi-Squared statistic of 2121.47 (compared to 5489.08 for Closheim's model) and a lower absolute error (mean absolute error: 23.75% and standard deviation: 16.06%). The difference in the cor-

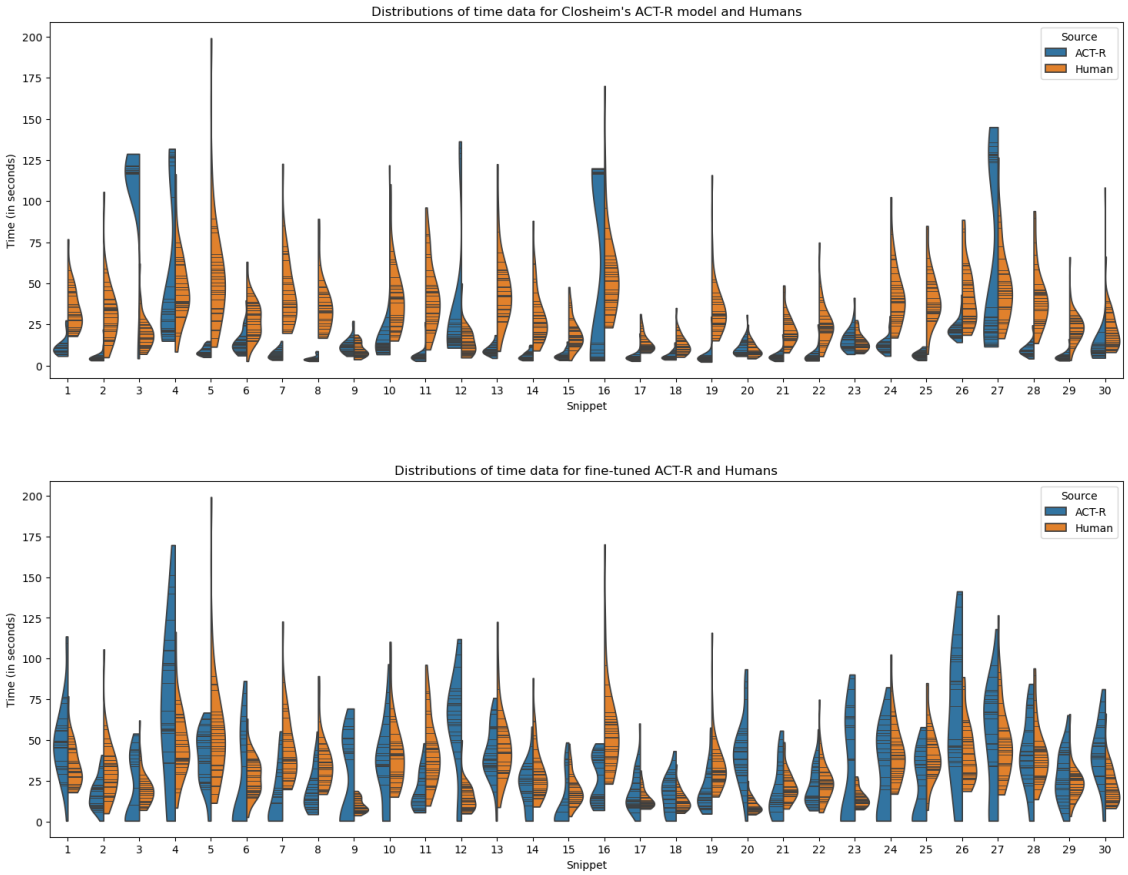

Figure 4.2: Distributions of answering time for the ACT-R model and human participants. Each violin represents the data for one snippet. The orange half (right) represents the empirical data and the blue half (left) represents the simulated ACT-R data. The data for the model by Closheim is in the top row and the data for the fine-tuned model is in the bottom row. Each line in a violin represents an individual data point.

rectness error between the two models can be seen in Figure 4.1 (bottom right). Regarding the type of errors, the same tendency as for Closheim's model in types of errors holds, with ACT-R stopping due to failed retrievals more often than it retrieves a wrong value (418 missing answers out of 537 wrong answers, 77.84% missing answers). One notable difference from Closheim's model is that the fine-tuned model makes more errors than programmers for most snippets, only making fewer errors on 3 out of 30 snippets, while the model with its original parameters shows no such clear tendency, having both snippets with significantly more and significantly less correct answers.

RQ0: The original model by Closheim shows shortcomings in the simulation of both correctness and answering time on a larger set of snippets. An ACT-R model with the same rules, but fine-tuned with data of the new snippets, shows better results for correctness and time

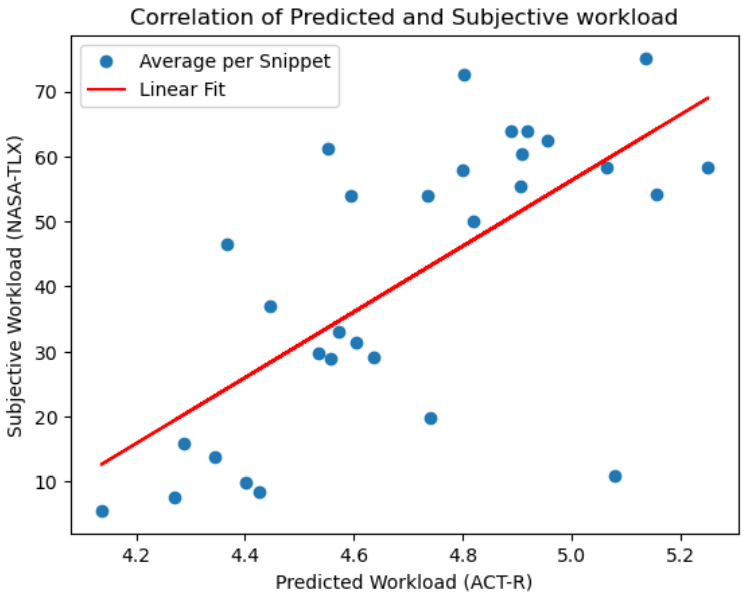

Figure 4.3: Scatterplot showing the relationship between the predicted workload and subjective workload. Each data point represents the average predicted and subjective workload for a single snippet. The red line shows the result of a linear regression between the two variables.

4.1.2 Research Question 1

We used the fine-tuned model for the remaining research questions, because it simulated the empirical data much better. Using the data from the fine-tuned model, we calculated the average predicted workload for each snippet based on the workload prediction by Jo et al. [39]. We then calculated the correlation between that value and the average subjective workload, given by the NASA-TLX answers of our participants, for each snippet using Kendall's tau. For the subjective workload, we calculated a general NASA-TLX score for each participant and snippet by averaging their answers to all three questions for that snippet. We then averaged over all participants for a snippet to obtain the average subjective workload for that snippet. Overall, we found a strong correlation between the two measurements (Kendall's tau = 0.503). Since we found a strong correlation between subjective and predicted workload, we next investigated if there is a linear relationship between the two variables, similar to the one found by Jo et al. By applying a linear regression to our data, we found the following linear relationship between subjective and predicted workload:

$$WL_{NASA-TLX} = 50.51 * W_{avg} - 196.26$$

with a p-value of 0.00004 (FDR-corrected = 0.0007), a root mean square error of 15.791, and a R^2 value of 0.629. This relationship is shown in Figure 4.3. Since we found a linear relationship between the average subjective workload and the average predicted workload, we decided to also investigate how well the distributions for each snippet matched between the subjective workload and the predicted workload with the linear regression applied. The comparisons for each snippet can be seen in Figure 4.4. We applied the Kolmogorov-Smirnov test to the data for each snippet to check if the distributions showed significant differences.

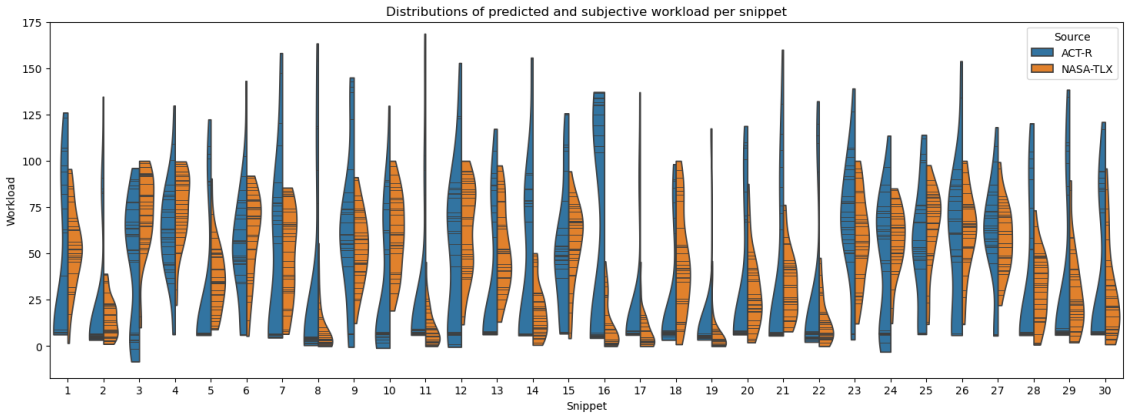

Figure 4.4: Distributions of predicted workload for the fine-tuned ACT-R model and subjective workload reported by participants. Each violin represents the data for one snippet. The orange half (right) represents the subjective workload data and the blue half (left) represents the predicted workload data. Each line in a violin represents an individual data point. The ACT-R data had the linear regression from Figure 4.3 applied to make comparing the data easier.

We found that only a few snippets (Snippets 9, 23 and 27) did not show significant differences between the distributions for predicted and subjective workload. Notably, for many snippets, the ACT-R workload prediction features a group of data points with very high workload prediction. These are runs in which the model made an error, which greatly affects the workload prediction. Additionally, the runs without errors were often all clustered close to each other at low values of predicted workload.

RQ1: The average of the ACT-R based workload prediction correlates strongly with the average subjective workload measured by the NASA-TLX, and the two measurements show a significant linear relationship.

4.1.3 Research Question 2

For analysing the ERP, we looked at the first line of each snippet and the last line before the print statement in each snippet. While we initially planned to also investigate the lines containing the print statements, we did not do so because 68.125 % of trials in these lines were excluded according to our artefact rejection rules. The most common reason for exclusion for these lines was that the line was shown for a time span shorter than one second, and therefore the end of the ERP would likely contain artefacts from the following activity (520 of 654 exclusions). For the first and last line, we removed trials from our data that were either too short or contained artefacts, as explained in our Methodology. To increase the amount of data points in an ERP, we grouped our snippets into easy, medium, and hard snippets based on the average NASA-TLX rating for each snippet. The eight lowest rated snippets made up the easy group, the eight highest rated made up the hard group, and eight medium rated snippets made up the medium group. Between the groups, we left 3

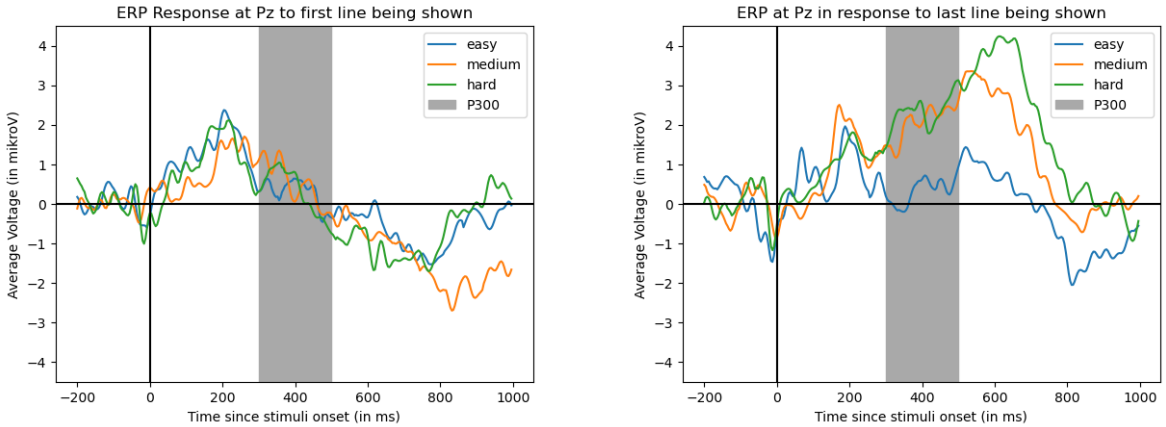

Figure 4.5: ERP responses to the first and last line of a snippet being shown, grouped by difficulty. The area relevant for the P300 amplitude is shaded in grey.

snippets as a buffer that were not assigned to any group, to prevent having snippets with very similar subjective workload in different groups.

For the first line (see Figure 4.5 left), in which all snippets should have the same workload, we found that no significant difference in mean P300 amplitude between groups exists (p-value = 0.932, FDR-corrected = 0.932). This supports the validity of using the mean P300 amplitude as a measure of mental workload, because it shows that the amplitude does not differ significantly when workload is equal. For the last line (see Figure 4.5 right), the Kruskal-Wallis test found a significant difference between groups (p-value = 0.021, FDR-corrected = 0.038). In the pairwise comparison of the groups, we found that the easy group showed significantly lower mean P300 amplitude than both the medium (p-value 0.015, FDR-corrected = 0.032) and hard group (p-value = 0.021, FDR-corrected = 0.038 for medium group), with no significant differences between the medium and hard groups (p-value = 0.721, FDR-corrected = 0.811). Notably the effect between easy and hard or medium snippets continued after the time window which we evaluated for the P300 amplitude (300 ms to 500 ms), and appeared to peak at around 600 to 650 ms based on a visual inspection of the ERPs in Figure 4.5.

RQ2: Easy snippets show a significantly lower P300 amplitude than medium and hard snippets, with no significant difference between medium and hard snippets

4.1.4 Research Question 3

As previously discussed, we investigated two EEG-based workload measurements for this research question: the ratio of frontal theta to parietal alpha band power and the mean P300 amplitude.

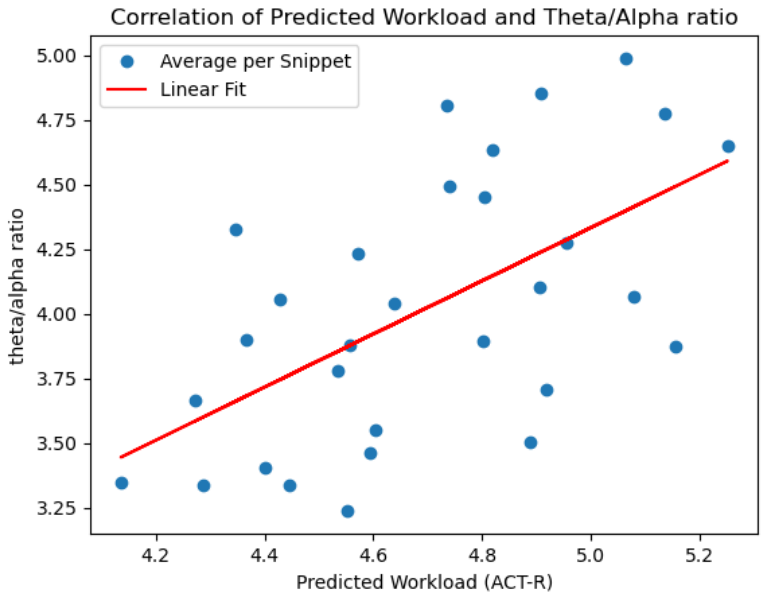

Figure 4.6: Scatter plot showing the relationship between the predicted workload and the theta-to-alpha ratio per snippet. Each data point represents the average theta-to-alpha ratio and predicted workload for a single snippet. The red line shows the result of a linear regression between the two variables.

4.1.4.1 Theta-to-Alpha Ratio

We found a medium correlation between the ACT-R based workload prediction for a snippet and the average theta-to-alpha ratio during comprehension of the snippet (Kendall's tau = 0.39). To investigate the relationship between the predicted workload and alpha-to-theta ratio, we applied a linear regression. The best fit was:

$$\frac{\theta_{frontal}}{\alpha_{parietal}} = 1.028 * W_{avg} - 0.806$$

with a p-value of 0.0007 (FDR-corrected = 0.007), a root mean square error of 0.413 and an R^2-value of 0.460. This relationship can be seen in Figure 4.6. Additionally, we investigated how well the distributions of predicted workload and alpha-to-theta ratio match by using the Kolmogorov-Smirnov test. We found that, for all snippets, there were statistically significant differences between the distributions. As can be seen in Figure 4.7, the most notable differences are that the theta-to-alpha ratio features far more spread out distributions than the subjective workload, and that the form of the distributions varies considerably for all snippets, with the predicted workload usually far more concentrated in a few spots, while the theta-to-alpha ratio is spread out over its entire distribution. This indicates that the ACT-R model can predict the average workload as measured by the theta-to-alpha ratio but cannot in its current form cover the large variety of human brain activity.

4.1.4.2 P300 Amplitude

Because we did find significant differences between easy and difficult levels of mental workload in the mean P300 amplitude, we investigated the correlation between predicted

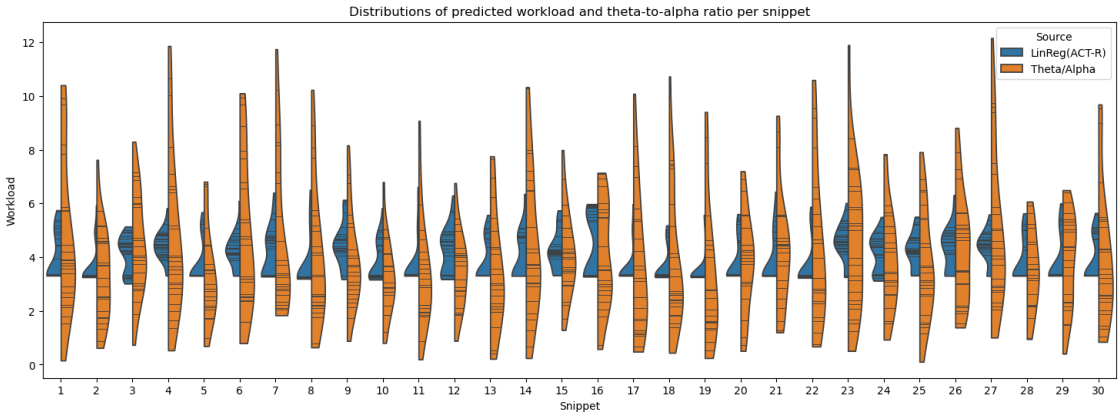

Figure 4.7: Distributions of predicted workload for the fine-tuned ACT-R model and theta-to-alpha band power. Each violin represents the data for one snippet. The orange half (right) represents the subjective workload data and the blue half (left) represents the predicted workload data. Each line in a violin represents an individual data point. The ACT-R data had the linear regression from Figure 4.6 applied to make comparing the data easier.

workload and the mean P300 amplitude for each snippet. For this evaluation, we could not maintain the groups of the previous evaluation. Since we already know that the group difficulty perfectly correlates with mean P300 amplitude, an analysis of the correlation between the predicted workload and mean P300 amplitude per group would effectively only analyse the correlation between subjective workload and predicted workload. Therefore, we have to analyse individual snippets for this task so that we have enough data points for the correlation, even though that limits the accuracy of each individual data point, as those then are calculated with a sample size that is smaller than the recommended 60 samples for analysing the P300.

Overall, we found a weak correlation between P300 amplitude and predicted workload (Kendall's tau = 0.149). We then applied a linear regression, which found the best fit as:

$$P300_{amp} = 1.163 * W_{avg} - 4.054$$

with a p-value of 0.342 (FDR-corrected = 0.410). As such, no significant linear relationship between the two variables exists. While a more detailed analysis of the ERPs for each snippet could reveal additional information regarding relevant components and their relation with increased workload, such an analysis would require significantly more data to be reliable. We had only 29 usable samples for a single line in a snippet before applying our artefact rejection criteria, while at least 60 samples are recommended for analysing the P300. As such, we did not conduct such an analysis because any results it may yield would not have been reliable.

RQ3: The average of the ACT-R based workload prediction showed a medium correlation with the ratio of frontal theta to parietal alpha band power and both measures have an approximately linear relationship. For the mean P300 amplitude we found only a weak correlation and no significant relationship. The distributions of the EEG-based workload measurements we explored show a much higher variance compared to the ACT-R based workload prediction

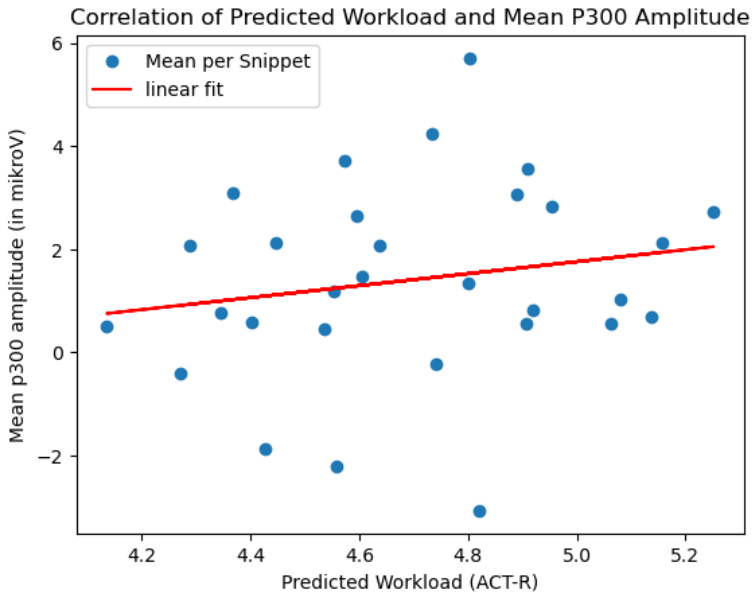

Figure 4.8: Scatter plot showing the relationship between predicted workload and mean P300 amplitude per snippet. Each data point represents the average predicted workload and P300 amplitude for a single snippet. The red line shows the result of a linear regression between the two variables.

4.1.5 Research Question 4

We investigated the overall correlation of theta-to-alpha ratio and predicted workload for individual lines. We found no significant correlation between the two measures (Kendall's tau = -0.041), with the theta-to-alpha ratio showing much greater variance over all lines while most of the data for the ACT-R model is clustered around the same value (see Figure 4.9).

Besides the relationship of the predicted workload and mental workload over all snippets, we also investigated the effect of different sources of mental workload on the workload present in a line. We first analysed the impact of increased working memory load. Of the previously discussed sources of mental workload, this covers primarily the amount of variable declarations and redeclarations. As increasing either of those is the main driver of snippet length, we used line numbers to group our lines by working memory load. We began with investigating the relation between the theta-to-alpha ratio during reading a line and the line number. In this, we restricted our statistical analysis to the first 7 lines, as only 4 out of 30 snippets had 8 or more non-print lines (and only 2 snippets had 9 or more lines), and as such, the averages in the later lines would be heavily influenced by the workload of individual snippets. For the sake of completeness, we still included the later lines in our figures and descriptions, but any effects seen in lines after line 7 should be treated carefully as they are susceptible to individual outliers. We found that the average

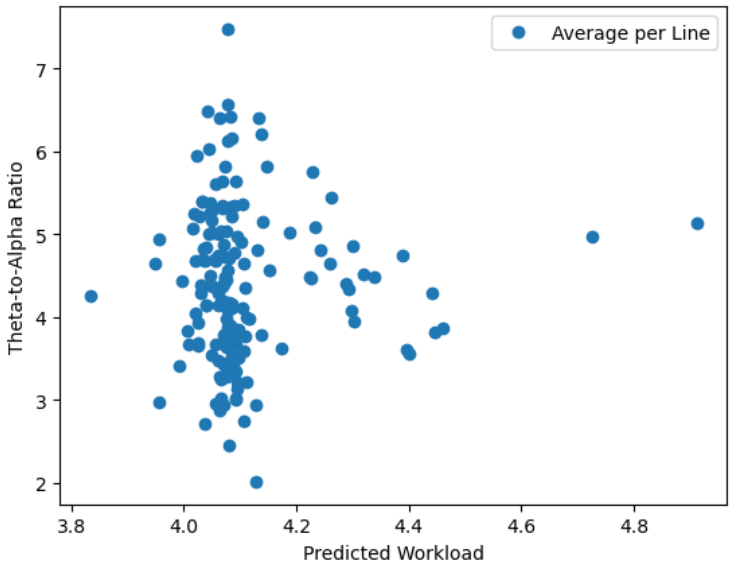

Figure 4.9: Scatter plot showing the relationship between the predicted workload and the theta-to-alpha ratio for individual lines. Each dot represents the average predicted workload and theta-to-alpha ratio for one line.

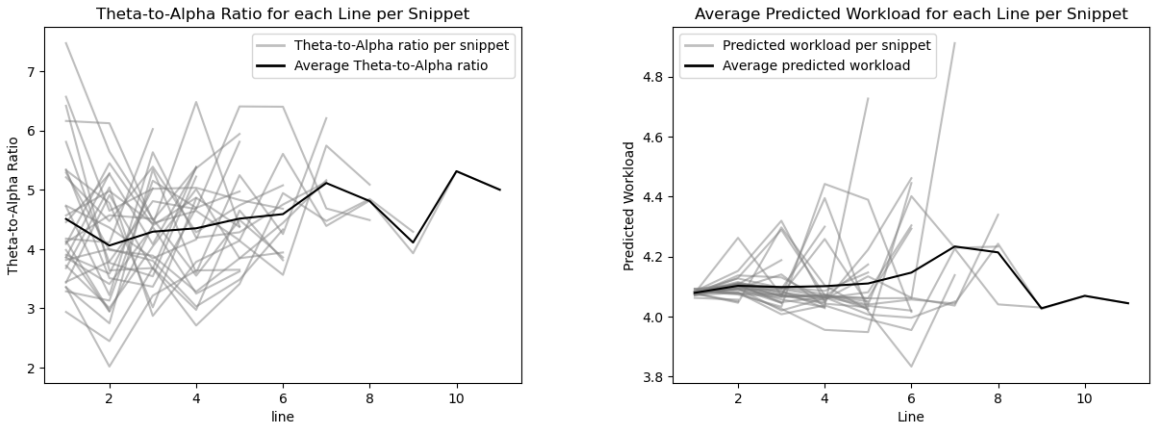

Figure 4.10: Theta-to-alpha ratio (left) and predicted workload (right) for each line in each snippet. Black lines are the averages over all snippets for the respective line numbers and each grey line is the data for one snippet

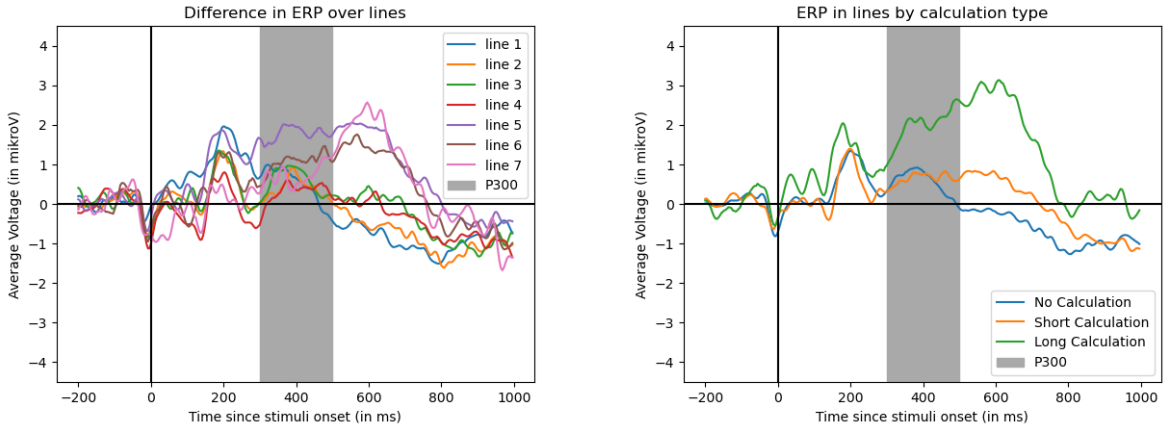

Figure 4.11: Grand average ERP waveforms for the presentation of specific line numbers (left) and for lines with different lengths of calculations (right). Each waveform is presented relative to the onset of the relevant line.

theta-to-alpha ratio increased with increasing line number up to line 7, dropping off until line 9 and then rebounding for the last two lines (see Figure 4.10. Overall, we found a very strong correlation between line number and the average theta-to-alpha ratio for the lines up to line 7 (Kendall's-tau=0.714). Notably, there are significant differences between snippets in a single line, indicating that working memory load may not be sufficient to explain the theta-to-alpha ratio.
To analyse the effect of line number on P300 amplitude we created ERPs for the presentation of lines with a specific line number. Here, we found no statistically significant differences in P300 amplitude between lines in the Kruskal-Wallis test (p-value = 0.070, FDR-corrected = 0.105), but a visual inspection of the ERP waveforms shows possible differences later on, at around 600 ms (see Figure 4.11). These differences take a similar form to the differences we found between the different groups in Research Question 2.
We also investigated if there was an effect of the line number on the predicted workload during reading the line. We found a similar behaviour to the theta-to-alpha ratio, with the workload increasing with line number up to line 7, where it then dropped off to the level of the few longer snippets that remained. However, unlike the theta-to-alpha ratio, the predicted workload did not rise significantly for lines 10 and 11. In our statistical analysis, we found a very strong correlation (Kendall's tau = 0.810) between the line number and the average predicted workload for lines of that group up to line 7.

To investigate the remaining causes of mental workload that are relevant to our snippets, we then compared the workload of lines with calculations of varying lengths. For this, we grouped lines into three groups based on if they contained no calculations, only calculations that contain exactly two operands or calculations with more than 2 operands. Notably, these groups were not balanced, with far more lines having a short (48) or no (98) calculation than lines with long calculations (15). For the theta-to-alpha ratio, we found no statistically significant effect of calculation length on the mean workload (p-value = 0.197, FDR-corrected = 0.273). The distributions of the theta-to-alpha ratio for snippets with differing calculation length can be seen in Figure 4.12 (top).

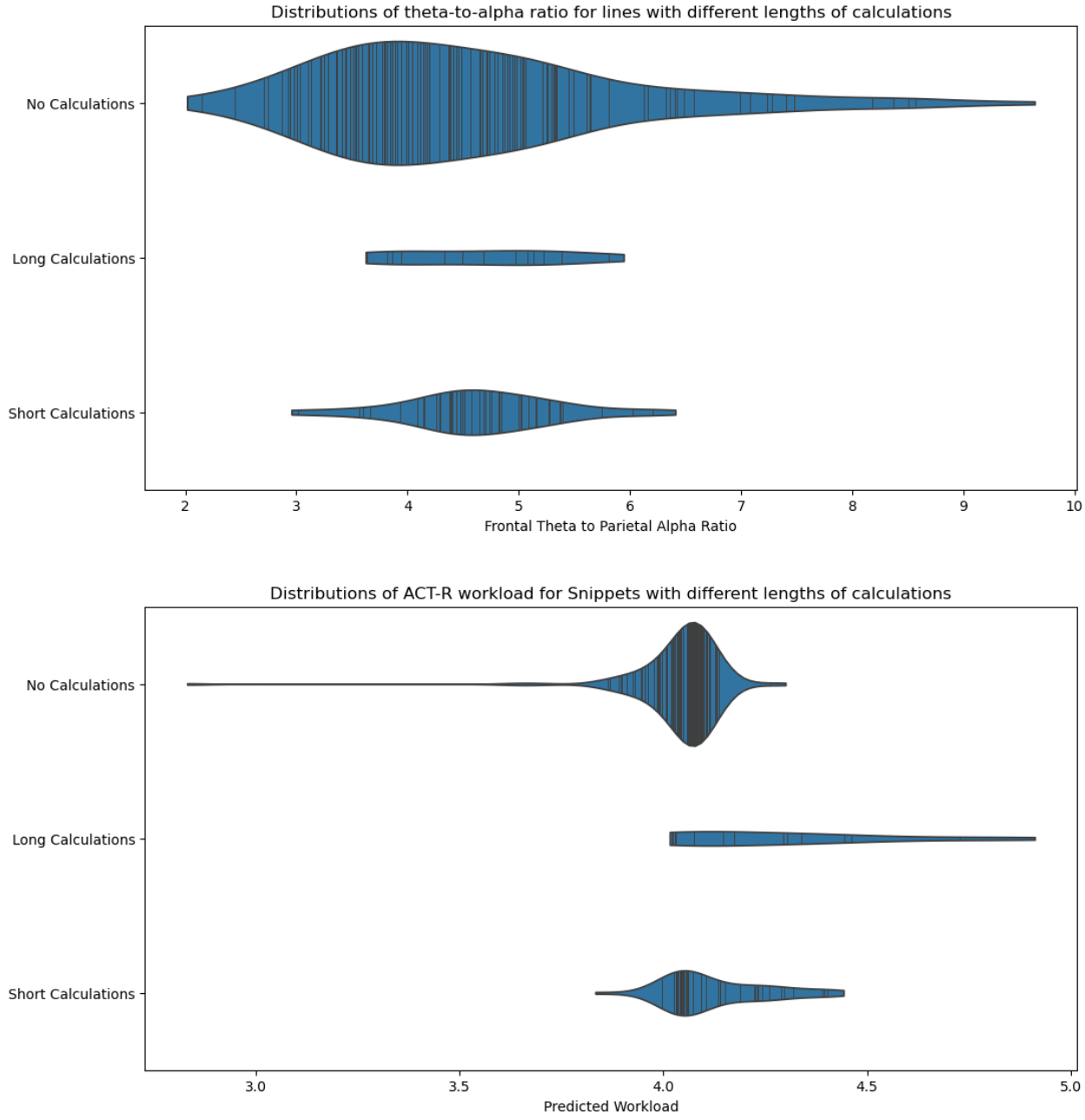

Figure 4.12: Violin plot of the distributions of theta-to-alpha ratio (top) and predicted workload (bottom) for lines with and without calculations. Each line in a violin represents one data point, with the area of a violin representing the amount of data points.

We also compared the mean P300 amplitudes between the three groups (see Figure 4.11) and found that a significant difference in the P300 amplitude between the groups exists (p-value = 0.0136, FDR-corrected = 0.033). The pairwise comparison showed that the significant difference is between lines with long calculations and those with no (p-value=0.006, FDR-corrected = 0.023) or short (p-value=0.005, FDR-corrected = 0.023) calculations, with almost no difference in P300 amplitude between the latter two groups (p-value = 0.920, FDR-corrected = 0.932). The difference again became more strongly visible in the later duration of the ERP, at around 650 ms, comparable to the effects we saw for different difficulties of snippets.
For the predicted workload based on ACT-R, we found that a statistically significant effect of calculation length on the predicted workload exists (p-value = 0.011, FDR-corrected = 0.032). The pairwise analysis found that no statistically significant difference between the predicted workload in lines with short calculations and those with long calculations (p-value = 0.271, FDR-corrected = 0.341) or no calculations (p-value = 0.066, FDR-corrected = 0.105) exists. However, between lines with no calculations and lines with long calculations, we did find a statistically significant relationship (p-value = 0.007, FDR-corrected = 0.023), with lines with long calculations having a higher mean predicted workload. The distributions of the predicted workload for snippets with varying calculation length can be seen in Figure 4.12 (bottom).

RQ4: The theta-to-alpha ratio of a line is strongly correlated with increasing working memory load, but there is no significant effect of increasing calculation length. In contrast, the P300 amplitude is primarily affected by calculation length, with no significant effect of increasing working memory load. The ACT-R workload prediction can predict the increase in average theta-to-alpha ratio per line but not the effect of calculations on workload and mean P300 amplitude. A prediction of the theta-to-alpha ratio or mean P300 amplitude on individual lines is not possible for our ACT-R model.

4.2 Discussion

In the following, we first discuss the performance of our fine-tuned model. Based on that discussion, we present possible improvements to the model and the fine-tuning process that could help align the model closer with empirical data in future work. We then analyse the effect we found in the ERPs around 650 ms in more detail and investigate what ERP component this effect could represent. Lastly, we discuss the insights we gained from our investigation of the prediction of mental workload with the ACT-R model and how the results of the workload prediction can be used to inform future design decisions regarding the model.

4.2.1 Model Performance

We found that a version of the ACT-R model by Closheim, fine-tuned to the program comprehension task and snippets we used, provides an improved simulation of both correctness and error rate, even though it cannot mirror the exact distributions of these variables in empirical data. The fine-tuned model shows a significantly better fit to the empirical data in its simulation, compared to the ACT-R model with the parameters found

by Closheim, which does not adequately simulate program comprehension on the snippets we investigated, most notably showing major deficiencies in simulating answering time. It follows that the parameters found by Closheim are not universally applicable to program comprehension. However, the newly fine-tuned model working as a simulation indicates that the overall idea for the model is likely valid and could work in a more general setting in the future. It is important to note that, while we found new parameters that work adequately for our snippets, additional work with new snippets is needed to investigate if the parameters we found do work for more general snippets and populations, or if their correctness, and consequently that of our model, is limited to the set of snippets we used. It seems likely that the exact parameters we found are not universally applicable either, and that future work will have to fine-tune a new model if it differs significantly from our study in either the type of snippets used or the population of participants employed in the study.

4.2.2 Optimization

A notable restriction for both our work and that of Closheim was the long run time of the fine-tuning of the parameters (roughly 5 days for 1000 iterations) and that it did not always converge in the time frames we had available (neither Closheim nor us were able to successfully optimize for both time and correctness at the same time). Future work could investigate alternative approaches to this optimization problem which may yield better and faster results. Aside from completely changing our approach to the optimization, it might also be worth investigating the use of a different optimization target. Especially for the optimization in time, this might be useful as the Kolmogorov-Smirnov statistic we used often reaches its maximum value (1 per sample) and this might adversely affect the optimization, as it means that often no new information is gained by the Bayesian optimizer when we are far away from optimal parameter values.
Lastly, it might also be worthwhile to check if the current approach to optimization does yield better results with increased computing power and time, as both Closheim and we were limited to home PCs, and executing the optimization for more iterations might lead to a converging result. This possibility is to some extent supported by the results of our optimizations: while it did not converge, and our optimal result occurred relatively early (sample 285 of 1000), the optimization curve does repeatedly show improvement up to a point close to our optimal result, only to then drop back to the worst value possible for the Kolmogorov-Smirnov statistic. This repeated pattern likely indicates the optimization algorithm searching for possible optima in different areas of our search space. As such, it is possible that one such future attempt yields better results than the best one we have found so far.

4.2.3 Potential Model Changes

In addition to potentially different parameters for the model, future work should consider changes to the underlying model to better match the simulation of program comprehension to empirical data. For this purpose, it is important to consider what might cause the differ-

ences between simulated and empirical data and also what causes variance in that difference. Considering what factors cause variance in the fit between empirical and simulated data is important, since potential issues with the model do not necessarily show as directly causing a mismatch between empirical and simulated data. It is instead possible that the model compensates for this specific issue in optimization, causing the data to match well for the cases where the problem is present, but to be misaligned in the absence of the problematic factor.

Since we found that the error in the prediction of ACT-R differs greatly between snippets in both time and correctness, we investigated if there are some shared attributes between snippets that might explain that variance. Based on a visual inspection of our data, we found that, the correctness error increased with increasing snippet length (from almost equal performance to ACT-R being significantly worse), while for answering time, the error decreased up to a snippet length of 8 (from ACT-R taking longer to a relatively close match between the data). For snippets with more than 8 lines, the time error then reversed, with ACT-R being significantly quicker than humans. We suspect that this is partially caused by ACT-R seldom finishing these snippets, because it is unable to recall a variable early on, and then does not have a rule available to continue execution. In some of these cases, a very short answering time of exactly 500 ms was recorded for the ACT-R model. We do not know what causes ACT-R to stop operating after such a short time, as we would expect the ACT-R model to at least run for the full duration of a failed recall. However, even when we exclude these cases from our calculation the effect of snippet length on time error is still visible. As such, the effect of increasing snippet length leading to ACT-R being quicker compared to human data is apparently not just caused by these possibly mistaken measurements, but also applies to the remaining measurements. The length of a snippet showed a medium negative correlation with the relative correctness error (Kendall's tau: -0.45), and a strong negative correlation with the relative time error (Kendall's tau: -0.61). This overall strongly indicates that the length of a snippet has an effect on how well the ACT-R model simulates the snippet.

The length of a snippet correlates with many different factors that all may impact the time and correctness of a snippet. We investigated three of those possible factors, which correlated to the primary sources of workload in our snippet: The amount of variables used in a snippet, the number of variable reassignments, and the amount of calculation steps necessary for a snippet. All of these factors showed strong correlations with the snippet length. Between the factors, we found that the number of calculations correlates with the other two factors, and that the numbers of variables and reassignments are not correlated with each other. The number of variables had no significant correlation with either error measurement, while the other two factors both showed correlations similar to the one we found for the snippet length. Notably, the number of redeclarations correlated even stronger than length with the correctness error, while the correlation for the number of calculations was weaker. Overall, this indicates that the variance in the correctness of ACT-R between different snippets is likely caused, to some extent, by differences in the amount of redeclarations in snippets. A future study that includes varying the redeclaration count while keeping the line length constant could investigate this in more detail, and confirm

```
x = 6
m = 4
c = 2
k = 8
a = x * c - k
print(a)
```

```
x = 3
a = 7
x = 6
a = a + x
print(a)
```

```
b = 6
n = 4
g = 5
b = 2
z = g - n + b
print(z)
```

```
j = 2
u = 3 * 2
q = 5 - j
w = u + 7
j = 6 - 4
s = w + j - q
print(s)
```

Figure 4.13: The snippets we used in the categorisation of errors. From left to right: 1, 2, 5 and 6.

if the redeclaration count is indeed a deciding factor for the variance in simulation error. For future work aiming to improve ACT-R this identifies a potential area of improvement for the model, although it remains important to identify what exactly is the cause of this discrepancy.

One potential cause is that the ACT-R model does not currently include any mechanism to reinforce the current, correct value of a variable after it has been declared or redeclared, relying fully on the effect of the more recent declaration on the activation of the corresponding chunk instead. By comparison, over half (67.6%) of our human participants indicated in the interview after the study that they kept mentally, verbally, or physically repeating the current list of variable assignments to help remember them. Most commonly, they reported repeating the list once or twice after reading each line, before they would continue to the next line. Implementing such a mechanism in the ACT-R model could allow it to more accurately mimic human behaviour, especially on more complicated snippets with many redeclarations, where, as previously explained, the model currently gives significantly more wrong answers and is faster than humans. Repeating the list of current values would allow the model to better remember declared and redeclared variables, while also taking longer for longer snippets.

Another possible contributing factor to the issues the model faces in simulating human performance is that it currently can only simulate two types of errors a human might make in the task we used. The model can either recall an old value for a variable or completely fail at recalling a variable and stop operating. In a qualitative inspection of human answers to four of our snippets (1, 2, 5, and 6, see Figure 4.13), we found multiple other possible error sources. We chose these snippets because they had a large variance in answers, allowing us to find different causes of errors. Notably, a basic inspection of the answers to other snippets shows that most other errors also cannot be sufficiently explained by recalling an old value for a variable. It is important to note that human participants were not instructed on a specific behaviour in case they could not recall a variable. Therefore, we cannot clearly decide what errors are caused by completely failed recalls and which have other causes. It is possible that participants simply entered arbitrary numbers or decided on something to answer in such a case for themselves. Future work that keeps the line-by-line restriction in reading snippets should instruct participants to give a specific, identifiable answer in such a case (e.g. an empty string) to exclude arbitrary answers after a failed recall and enable comparisons with the ACT-R model. Additionally, we recommend modify-

ing the ACT-R model to follow a similar behaviour, skipping through the snippet to give a specific answer in case of a failed recall, to closer match the human participants in such cases

Regarding the types of errors, our qualitative analysis identified the following types of errors:

- **Wrong variable access:** The value of a wrong variable is used in a calculation. One example of this is answering 0 to snippet 1, due to using the value of m instead of x in the last calculation.
- **Wrong operation error:** A wrong mathematical operation is used in a calculation. One example of this is answering 1 to snippet 2 due to calculating $a - x$ instead of $a + x$.
- **Redeclaration error:** An old value for a variable is used after that variable has been assigned a new value. One example of this is answering 10 to snippet 2, due to using 3 as the value of x instead of 6.
- **Partial calculation error:** Only part of a calculation is finished, discarding the remaining steps. One example of this is answering 1 to snippet 5, due to only calculating $g - n$ instead of $g - n + b$.
- **Calculation error:** The result of a calculation is incorrect due to an arithmetic error. Most commonly these occurred in additions and subtractions with carry-over, typically giving a result two higher or lower than the correct one. A secondary case of this error were off-by-one errors were a result differed by one from the original result. These occurred more generally on any calculation without a clear tendency as to what calculations were affected.
- **Combined errors:** Two or more of the above errors happen in combination. For example, a participant used an old value of the wrong variable in a calculation. An example of this is answering 9 to snippet 2, which is caused by using the old value for b as the value of a in the calculation of the result (while using the correct value of b for the actual variable b).

It is important to note here that we did not collect step-by-step protocols of participants' mental calculations (and instructing participants to protocol their steps in a written or verbal form would likely affect the steps they take), As such all of the error types mentioned above are based on conjectures we drew based on the answers participants gave. While this introduces some uncertainty into this analysis, we believe that the types of errors mentioned above occurred often enough, even in only the 4 snippets we investigated that they likely are actual error causes.

In addition to the previously discussed possible changes to the model and our optimization approach, future work should explore an optimization with more data, to enable a more accurate model. One way to do so would be to combine our data with that by Closheim [18] and Dörrzapf [23], who both used almost the same task in their studies. A notable roadblock to such an approach are the existing deviations between these studies, with the studies by both Closheim and Dörrzapf being online studies and our study being an in-person study

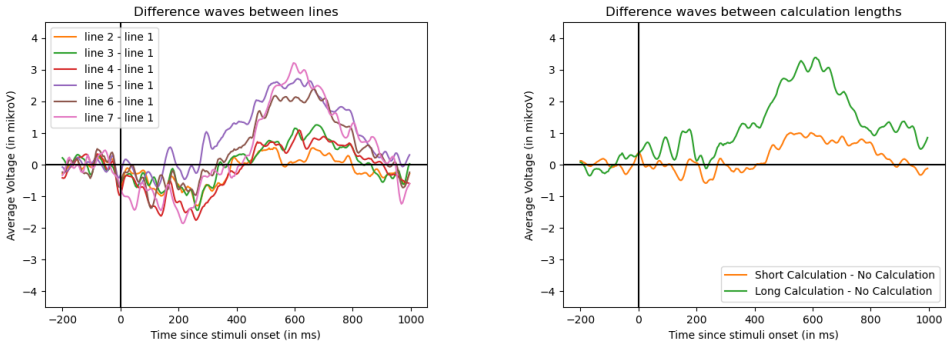

Figure 4.14: On the left: Difference waves between line 1 and lines 2 to 7. On the right: Difference waves of ERPs between lines with short and no calculations (blue) and between lines with long and no calculations (orange).

that also used EEG. This difference could cause differences in the behaviour of participants that might mean that a single model is not appropriate to model humans in both the online and the in-person setting. To investigate if this approach might be feasible, future work could compare some of the snippets present with only minor differences in all three studies. If, for these snippets, the data by all three studies is found to be in the same distribution, the effect of the setting and population on the results is likely negligible.

4.2.4 ERP in Program Comprehension

In our evaluation of RQ2, we found a clear difference between the ERPs of easy snippets and those of medium and hard snippets. While this difference was significant in the area we investigated for the P300 amplitude, it does not appear to be solely caused by the P300 component, as the effect carries on up to around 700 ms, significantly longer than a typical P300, and even increases further after the time period we investigated for the P300. Additionally, no clear P300 peak was detectable, which may indicate that the P300 overlaps with some other component to such an extent that it is not visible in the ERP, or that there is no P300 component present at all, with the entire effect being the result of some other component.
To gain further information about what causes this effect, we investigated the effect of increasing line number (and thereby working memory load) and calculation length in our evaluation of RQ4. As previously explained we restricted our analysis to lines 1-7 since only a few snippets are long enough to analyse further lines (4 for line 8 and 2 for line 9), which would greatly reduce the reliability of the EEG data. Figure 4.14 shows the differences in the ERP responses between line 1 and the later lines. The original ERPs for each line can be found in Figure 4.11. Overall, we can see that the primary difference in the ERP between lines occurs roughly between 500 and 700 ms, with the rest of the ERP being very similar across all lines. The main exception to this is an early difference at around 200 ms which differs between lines with no clear pattern. As to the differences between ERPs between 500 and 700 ms, we found that the amplitude in this time grows with higher line numbers. We can roughly identify three groups of lines with similar amplitudes, lines 1 and 2 as the

lowest, lines 3 and 4 slightly above those and lines 5, 6, and 7 with a significantly higher amplitude. Notably the differences between lines are the clearest around 600 ms, matching with the peak in difference between easy and medium or hard snippets.

Initially, this might be interpreted as the ERP response being significantly affected by the increasing working memory load with later lines, but when we investigate these groups of snippets, we notice that, with increasing line number, the ratio of lines with calculations in the snippet also increases. The way the ratio changes roughly aligns with the groups of lines we found to behave similarly. This means that, while the difference between later lines is possibly caused by differences in working memory load, it could instead also be caused by differences in the ERP responses to calculations compared to the responses to declarations or redeclarations.

In comparing the ERPs for declarations, short calculations, and long calculations across all lines, we found that there is a difference between these lines, with the lines without calculations featuring a peak at around 400 ms, aligning with our expectations of a P300 component, while the lines with short calculations show more of a plateau from 400 to 600 ms, reducing in amplitude afterwards until it is at the same amplitude as lines without calculations at around 800 ms. This plateau has roughly equal amplitude to the P300 peak in lines without calculations. Outside of this difference, both types of lines show very similar ERPs, as can be seen in their difference wave in Figure 4.14. By comparison, longer calculations showed both an overall higher wave in the ERP at around 400 ms and later increases further in amplitude up to a peak at around 600 ms.
This peak likely explains the peak we found in the last line for medium and hard snippets at around 600 ms (for medium snippets) to 650 ms (for hard snippets), as these groups often have longer calculations in the last line. Specifically, for both the medium and hard group 6 out of 8 snippets have a calculation with more than two operands in their last line, while none of the 8 easy snippets do. This high peak for difficulty might be part of the same effect that causes the plateau for calculations in general, as it peaks around the same time as the plateau begins to drop off. This means the plateau could be formed by inference between the P300 component and this second peak, which happen to be roughly equal in amplitude for calculations with two operands. Alternatively, it is possible that in lines with calculations there is no P300 at all, with the plateau we found in short calculations being its own component that simply happens to overlap in amplitude and timing with the P300 in lines without calculations.

Overall, our findings indicate that some component, other than the P300, influences the ERPs and is at least partially responsible for the differences in the ERPs between the different difficulty levels. Based on our results, such a component would have to affect the ERP starting around 350ms at the earliest, and continue to at least around 700ms; be present at Pz; and either be a positive component that grows in amplitude with increasing workload or a negative component that has a higher amplitude with lower workload. The rough outlines of this component for short and long calculations can be seen in the difference waves in Figure 4.14. In the case of a negative component growing with lower workload the difference waves would be inverted (e.g. the difference between lines with no calculations

and short calculations). It is also likely that this component is related to activity caused by calculations, and in that case, it would change amplitude with longer or more difficult calculations. One component that matches this description is the late positive slow wave, that has been reported in multiple ERP studies dealing with mental arithmetic [34, 37, 59]. This wave has been found to begin around 400 ms and continue for a varying amount of time though at least up to around 800 ms after stimulus onset. The slow wave has been found to occur for addition, subtraction, and multiplication tasks. To our knowledge, no study so far has investigated ERPs in division tasks. The amplitude of the slow wave has been linked to the problem size effect, a common effect in mental arithmetic that describes the fact that humans take longer and make more errors when engaging a problem with larger operands [84]. Studies of the problem size effect, both with and without ERP, have focused on differences in operand size in calculations of equal length, and so far, there has been no study investigating the effect of calculations with differing numbers of operands on the problem size effect. It is possible that a similar effect to increasing operand sizes also occurs when increasing the amount of operands. For the slow positive wave, a higher amplitude has been reported with increasing problem size [59]. With regards to spatiality, the slow wave has been reported in different studies to appear spread out over most of the brain (including parietal regions), restricted to only frontal regions or restricted to only parietal regions. It has been theorized that this difference is caused by variations in how well a task is memorized, with a more frontal focus indicating conscious effort in the calculation, and a more parietal focus indicating automatic processes, such as recalling well-known calculations [37]. Overall, we can see that the positive slow wave observed in mental arithmetic studies occurs around the same time as the peak in amplitude we observed for longer calculations in our study, occurs at least partially in the parietal region (e.g. at Pz) and has higher amplitude with more difficult calculations, matching with the description of the component we are investigating. As such we find that the positive slow wave is a possible explanation of the effect of increasing calculation length we observed, as it matches with the component we theorized in time, length, location and potentially cause. It is, however, important to note that our study was not designed to investigate the presence of such an effect on the ERP, and as such, there are some potential confounding factors, such as the location of the calculations in the code, that we did not control between calculations of different lengths. A further, more controlled investigation into the effect of the amount of operands on the positive slow wave could provide more clarity and potentially confirm or refute that the component we found is indeed the positive slow wave.

Another potential explanation for our findings is the P600 component, an ERP component that has mostly been investigated in the context of natural language. There, it is strongly associated with syntactic errors, such as a mismatch between subject and object in a sentence, or a disagreement between tenses [30]. The P600 is a positive wave focused in the posterior brain regions that peaks at or slightly after 600 ms. Interestingly, the P600 has also been observed in music [64], where it is caused by a mismatch in the keys of chords. Based on this, Patel et al. theorized that the P600 is not specific to natural language, but can occur in any rules-governed sequence, so long as the observer knows these rules consciously or subconsciously [64]. Notably, both arithmetic calculations and programming languages are such rules-governed sequences, and as such may elicit a P600 if a mismatch occurs.

For number sequences it has been shown that a wrong number completing the sequence does also a P600 [58]. Another important finding regarding the P600 is that it also occurs in sentences that, while syntactically sound, are semantically difficult to understand such as garden path sentences (sentences that change 'direction' during the sentence) or complex sentences that involve recalling previously used phrases to interpret correctly [27]. This shows some similarity with longer calculations, where it is necessary to recall previous intermediate results to obtain the final result of a calculation. As such, it is also possible that the component we observed is in fact a P600 component, caused by the need for recalling both previously defined variables and intermediate results of calculations. Additionally, the P600 might be affected by longer calculations being rarer than both single-operator calculations and simple declarations in our programs, and as such, might elicit a P600 component not due to the specific calculations, but because the longer multi-operator calculation represent a violation of the expected sequence, since they differ from other stimuli as an oddball stimulus. So far, only a few studies have dealt with the P600 in mental arithmetic at all and those that did were constrained to the field of number sequences and violations of the rules of these sequences. Additional work regarding the presence and possible causes of the P600 in mental arithmetic is required to identify if the component we found is indeed a P600.

Another component occurring around the same time as the P600 in natural language processing is a late frontal positivity that has been observed in sentences featuring a word that is unlikely, but still semantically and syntactically fits in the context [76]. An example of such a sentence would be: *'On his vacation he got some much needed sun'*, where *sun* is a less likely completing word for the sentence than *rest* would be. A recent study investigating fixation-related potentials in a program comprehension setting found this effect to also occur for confusing source code when the confusing part of the code is fixated. They found a late positivity that was strongest in the frontal areas of the brain, but affected all brain regions [9], including the parietal area which we investigated in our study. The possible component we described earlier occurred at the same time and with roughly the same amplitude at Pz, compared to the late positivity found in this study. Further similarities or differences between these three components and the one we described could be revealed by a future topographical analysis, investigating the distribution of the effect we found over the scalp and comparing the results with the known topographic distributions of the P600, late frontal positivity and positive slow wave components.

One other possible cause for our result regarding longer calculations is that the component we observed is not a single component, but is formed by the overlap of multiple components that are elicited by different parts of the stimuli. It is possible that participants look at different parts of the longer calculation in succession, and in such a case, each of these fixations and its associated processes would cause its own potentials. The overlap of these potentials could then form the component we have found in our study. Future work could make use of eye-tracking data as an additional modality to investigate if such an overlap between the fixation-related potentials exists, and if so, if it could explain the component we found.

To summarise we found a component in our ERP data that occurs as a positive wave

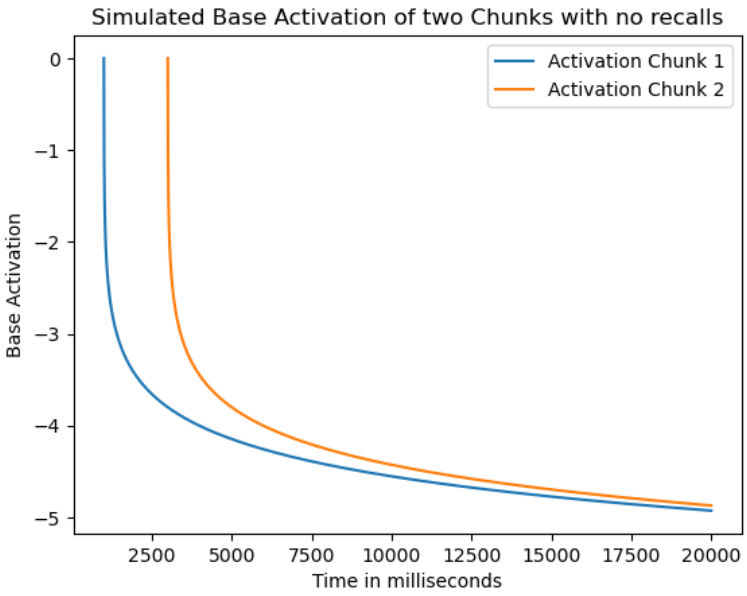

Figure 4.15: Simulation of base activation for two chunks in ACT-R. The first chunk (blue) is declared at 1000 ms after the model starts working and the second chunk is declared at 3000 ms. It is apparent that the difference in activation between the activation of the two chunks decreases the longer they are in memory without an access. This implies the longer we wait before a recall the more likely the model is to recall the wrong chunk.

roughly from 400 to 800 ms after stimulus onset and is affected in its amplitude by the amount of operands in a calculation. We found a number of possible explanations for this component. Firstly, it might be the positive slow wave that has been associated with differences in operand size. This would indicate that the positive slow wave is also affected by operator amount, or perhaps more generally by problem difficulty. Secondly, it might be a P600 component caused either by the increased complexity and necessity of more recalls in longer calculations or by the longer calculations representing a type of oddball stimuli as they appear less often than other stimulus. Thirdly, the component we observe might be the result of a late frontal positivity caused by the code with calculations being more confusing than code without calculations. Lastly, it is also possible that we did not observe a single component, but instead the interference of multiple components caused by fixations on different parts of the calculation and the mental processes associated with these fixations.

4.2.5 Workload Prediction

As previously discussed, we found that the ACT-R based workload prediction correlates strongly with subjective workload. This indicates that using ACT-R to predict workload in a program comprehension setting is generally possible. We also found a linear relationship between the predicted workload and the subjective workload. While this relationship matches the findings by Jo et al. in the type of relationship, it differs significantly in the exact variables. We assume that this is the case because there is no universal set of parameters for matching predicted workload on subjective workload over all possible tasks. One reason

why the existence of such a set of parameters appears unlikely is that studies have shown that the way a study is conducted can affect the NASA-TLX rating of a task. One example of this are anchoring effects, where the rating of a previous task affects the rating of the following task [19].

In analysing the workload prediction for individual snippets, we found that one snippet is a significant outlier from the relation between predicted and subjective workload. The predicted workload for snippet 16 is far higher than the average subjective workload for the snippet. We suspect the reason for this is that the snippet contains two variable redeclarations immediately after the original declaration. This apparently poses no large problem to humans, as they achieved a very high correctness for this snippet (96.875% correct). We assume that, when two assignments for the same variable directly follow each other, humans immediately discard the earlier assignment after finishing the later one, thereby minimising the risk of misremembering a value. In comparison the ACT-R model keeps both assignments in its declarative memory, which causes it to often fail recalls of the redeclared variable (46.67% of attempts for this snippet) or recall the old value of a variable (20% of attempts for this snippet). Additionally, the declarations being close to each other may make it even more difficult for the ACT-R model to recall the correct value. The activation value of a chunk decays over time and chunks stored at a similar time will have similar values when a recall is attempted, with the difference getting smaller as more time passes. An illustration of this behaviour in an artificial example can be seen in Figure 4.15. The previously proposed repetition of the current variable values could help fix this problem. Repeating the current correct value would lead to an increased activation for that value. This would in turn make the model more likely to recall the value correctly when it needs to access it.

When investigating the results of the workload prediction in greater detail, we found two more notable areas where the workload prediction could be improved. Firstly, we found that the difference in predicted workload between runs with and without failed retrievals within a single snippet is very large, so that much of the difference in predicted workload between snippets is apparently caused by failed retrievals. To some extent, this does match with the subjective workload data, where participants that answered incorrectly tended to rate the snippet as more difficult. It is, however, notable that there is a much clearer cut between the two cases in the predicted workload, while there is a more continuous data distribution for the subjective workload (see Figure 4.4). This might indicate that the impact of a failed retrieval on the predicted workload is too high, and that ACT-R runs with a failed retrieval should have a workload closer to those without a failed retrieval. This effect is likely caused by failed recalls always taking the longest time possible for a recall. Combined with the model stopping immediately after a failed recall, this greatly increases the impact the error term has on the average predicted workload. Future work could investigate alternative ways of applying the error term to the ACT-R functions. One way to do so could be to apply the error term not to the entire time of a failed recall, but only to a later portion. This would reduce the overall impact of an error on the predicted workload and also align the workload prediction closer to a human understanding of workload, as a human does not know they will fail at remembering something the moment they start the recall attempt.

The second issue we found in the workload model is that, since the model takes the average workload over time, and almost all activity happens during recalls, where only one module is active, the workload was very similar across snippets, even though the model might need a lot longer for one snippet compared to another. Future work that wants to investigate the ACT-R-based workload prediction might have to modify the workload prediction to account for most of the model activity happening in a single module. The workload prediction by Jo et al. is not well equipped to handle this, as it uses the activation of different models as its main measure of workload. One possible way to address this issue in the model could be to modify the workload prediction algorithm in a similar way to the modifications proposed by Oh et al. They incorporated the workload caused by previous subtasks as part of the workload of the current task by utilizing a decay function. The workload at a specific moment was calculated as the sum of the current workload, as it is calculated by Jo et al., and the sum over the average workload of each previous task, scaled by a decay function based on the time that passed since that task was finished. A similar approach, employed on a line-by-line or moment-by-moment basis, could allow for a workload prediction that differs much better between snippets and could simulate a much higher variance of workload than is currently possible. Another possible approach could be to modify the workload prediction to account for an increasing workload with continuous activity in a module. This would allow the workload prediction to differ between different lengths of recall, so that a longer recall would cause more mental load than multiple short recalls of the same total length. This would account for the fact that longer recalls represent a lower activation and therefore a chunk of information that is harder to recall.

Another simplification in the current model that might contribute to the relatively low differences between snippets is that all calculations are treated the same by the model, with the results of $1 + 1$ and $7 + 6$ having the same fixed activation in the declarative memory. Therefore, both are recalled with the same probability and duration when a request for the calculation is made. This goes against the previously mentioned problem size effect, which would predict that a calculation with larger operands takes longer to complete and applies even at the small magnitude of numbers the model works with [84]. To account for the problem size effect in calculation duration, the model could be changed so that larger calculations have a smaller initial activation, making the model take longer to remember the results. The distribution of activation between different operations would depend on which of these operations are more error-prone and take longer to perform for a human. To also account for the effect on correctness, it might be possible to use the spreading activation mechanic in ACT-R so that recalling a calculation also spreads some activation to similar calculations, allowing for the simulation of calculation errors in ACT-R. The exact form of this spreading activation would need to be based on a detailed analysis of the types of errors participants are likely to make in calculations. An additional problem in the model's handling of calculations is its treatment of calculations with more than two operands. Whereas humans have to remember partial results in those calculations and might misremember those values after accessing a new variable for the next step, the model stores partial results in its goal buffer, allowing immediate and error-free access. A possible approach to simulating this would be to change the model so that it stores partial results in

declarative memory or the imaginal module, instead of the goal buffer. This would cause the model to require additional time to recall partial results and make it possible for it to fail at recalling them.

4.2.6 ERP and Theta-to-Alpha ratio in workload prediction

We found that the ACT-R-based workload prediction correlates with the alpha-to-theta ratio, with a linear relationship between the two variables existing. This indicates that the ACT-R model could be used not only to predict subjective workload, but also to directly predict mental workload. Notably, we did not find a significant relationship between the mean P300 amplitude and the predicted workload for individual snippets. Since we know that the P300 amplitude does correlate with the reported subjective workload, this indicates that there is some aspect of workload in our snippets that does affect the ERP, but that the ACT-R model and the workload prediction algorithm do not account for. Additionally, we found that while both mean P300 amplitude and theta-to-alpha ratio show significant correlation with subjective workload, they do not correlate with each other. This shows that P300 amplitude and theta-to-alpha ratio likely represent two different aspects of mental workload. Since only the theta-to-alpha ratio correlates with the predicted workload, it is likely that the part of workload measured by the ERP is largely missed by the ACT-R model. Based on our findings in RQ4 the ERP response and mean P300 amplitude is likely strongly affected by calculation length and problem size, while the theta-to-alpha ratio is primarily affected by working memory load.

As previously explained, we found a later component that is affected more strongly by the calculation difficulty than the P300. Therefore, we also investigated the relation of the mean amplitude between 500 and 800 ms with subjective and predicted workload. We found a medium correlation with subjective workload (Kendall's tau = 0.398), and only a weak correlation with the theta-to-alpha ratio (Kendall's tau = 0.205) and the predicted workload (Kendall's tau = 0.287). It is important to note that longer snippets more often featured longer calculations, and therefore some correlation between working memory load and workload introduced by calculations is to be expected. Future work could bring more clarity here, as it could focus on varying calculation difficulty while keeping working memory load stable and vice versa.

The workload prediction largely missing the workload induced by calculations is further reinforced by the model showing a lower average predicted workload for lines with short calculations than for those without. These mismatches between workload prediction and actual subjective workload reinforce that the model treating all calculations as the same difficulty, and having immediate access to all partial results by using the goal buffer, is a significant issue in the ACT-R model. Fixing this issue would likely not only improve the workload prediction, but also allow for a better simulation of both behavioural data and time.

Our second EEG-based measurement, the theta-to-alpha ratio, grows with growing line

number, but shows no significant effect of longer calculations, implying that it is mainly modulated by working memory load, with no clear effect of more difficult calculations in a single line. This aligns with the findings in literature that the theta band is primarily related to memory access. To more conclusively test if an effect of calculation difficulty in a line on theta-to-alpha ratio exists, future work could use snippets only varying in the difficulty in a single line to enable a more controlled experiment.

Overall, we found that using ACT-R to predict workload during program comprehension tasks is feasible and could, with an expanded ACT-R model that can process more varied code, potentially become a valuable tool for analysing code complexity. However, it might be necessary to make some modifications to the workload prediction algorithm to account for the specific types of activation caused by program comprehension tasks. We also found that the workload prediction can be an important tool for finding issues in the ACT-R model, as we were able to identify problems in the model and propose potential solutions to these problems based on mismatches between the workload prediction and subjective workload data. Future work aiming to expand the ACT-R model to handle more complex programs should incorporate the workload prediction in its evaluations, because it may point towards mismatches between simulated and empirical data that the analysis of behavioural data does not reveal.

4.3 Threats to Validity

In this section, we discuss threats to the validity of our results and steps we took to mitigate these threats in our study design and evaluation.

4.3.1 Construct Validity

One variable in our study that might be affected by threats to construct validity is the use of the theta-to-alpha ratio as a measure of mental workload, as there exist many other frequency band power based measurements that are used as measures of mental workload in different contexts and for different tasks. As including all, or a large selection, of these measures would lead to issues with an increased rate of false positives, we decided to only use one specific band power based measurement of mental workload. To limit the effect on construct validity, we chose a measurement that has been successfully used in multiple earlier code comprehension studies investigating mental workload. In this way, we could ensure that the measurement we used has been shown to measure mental workload in program comprehension. Nevertheless, we cannot be sure that no better measurement of mental workload exists that might have been more appropriate to use. It might, for example, be the case that the theta-to-alpha ratio only measures specific aspects of workload that are present in code comprehension studies, but some other aspects that are also important are missed by it.

A second measurement that faces similar issues is the use of the mean P300 amplitude for measuring mental workload using ERP responses. While the P300 amplitude was used in studies using the n-back task and some similar tasks, other components have also been

shown to vary with changes in workload. In fact, based on our analysis, it appears possible that one such component, the slow positive wave that has been found in studies dealing with mental arithmetic, has a significant impact on our data. We based our initial assumption of the P300 being usable as a measure of mental workload on previous studies showing the effect of a reduced P300 with increasing workload in tasks similar to the one we used in our study.

4.3.2 Internal Validity

We employed common strategies to limit the impact of many threats to internal validity. For example, we used a different pseudo-random order of the snippets for each participant to cancel out potential ordering effects, and included breaks in our experiment design to allow participant's mental workload to reset. Additionally, we averaged our workload measurements over many participants to account for the high amount of noise and inter-subject variance that is present in EEG data. To further reduce the impact of noise on these measurements, we applied robust outlier removal approaches that were tailored to the specific weaknesses of the two different approaches.

Nevertheless, some threats to internal validity remain, most notably, the change we measured in P300 amplitude appears to likely be caused, at least partially (and perhaps completely), either by a change in a later component or the inference of multiple components. For a more in depth discussion of this aspect, see the section on ERP in program comprehension in the Discussion.

Lastly, it is important to note that the ACT-R based workload prediction is reliant on the quality of the underlying ACT-R simulation. To account for this threat, we decided to fine-tune the model by Closhiem due to the discrepancies between its simulated data and the empirical data. However, since even our newly fine-tuned ACT-R model shows some differences from the human data, it is possible that issues in the ACT-R model of code comprehension limit the validity of our findings. For this study, we were unable to further mitigate this issue, as ensuring that the ACT-R model can simulate all relevant aspects of code comprehension and can make all errors commonly made by humans would require significant alterations to the model and additional studies to ensure the validity of the changes. While this would far exceed the scope of this thesis, we have outlined potential ideas that future work could use to modify the model and reduce the difference between model and empirical data. An additional aspect of the ACT-R model that might affect the validity of our findings are the runs of the ACT-R model that terminate earlier than we expected to be possible. We do not know what causes the model to terminate so early, and thus cannot clearly say if this represents an issue in the modelling, an error in model execution, or another unrelated issue. Therefore, we cannot determine if these runs pose an issue for our workload prediction or our evaluation of the model.

4.3.3 External Validity

As our study was primarily intended to prove the overall feasibility of the ACT-R based workload prediction in a program comprehension setting, there are multiple significant threats to its generalizability. One such threat is that all of our participants were students, with most of them being enrolled in a master's program. Given such a narrow participant pool it is possible that our findings do not generalize to a larger population, such as all professional programmers. Even if the overall findings hold, the ACT-R model would likely need to be fine-tuned with data fitting to a specific level of proficiency, to be able to adequately simulate the behavioural data of programmers at that level.

Since the potential causes of workload in our snippets were correlated with each other, it is difficult to be certain what exactly affects a given measurement of mental workload. Specifically, the correlation between longer snippets and larger calculation sizes means that it is difficult to investigate these two causes independently from each other. One example where this issue possibly occurred is in the differences in the P300 amplitude in the different lines. It seems likely that the differences we observed were caused by more and longer calculations being present in later lines, and not by an increasing working memory load in later lines. Regrettably, it is not possible to conclusively determine the cause of the effect, as an analysis comparing groups with long and short calculations in both early and late lines fails due to the lacking sample size of some groups, as we, for example, have few snippets that are short but feature long calculations. As outlined earlier, a study designed specifically to investigate the impact of different causes of mental workload is needed to clear up the exact relationship between different causes of mental workload and measures of mental workload.

One last limitation of our study, compared to a real life development setting, is that participants were instructed to move as little as possible, to limit movement artefacts in the EEG. As a consequence, it is possible that some potential strategies to complete the program comprehension tasks, such as forming the numbers to be remembered with their fingers, were avoided by some participants to prevent additional movement. In a setting without an EEG recording, it is possible that some participants would have used different strategies and therefore could have experienced a different workload and different answering time and correctness.

4.3.4 Conclusion Validity

Some of our evaluations regarding the ERP responses of participants are of limited statistical power, as we had only a limited sample size for them. For example, each point in the correlation between P300 amplitude and predicted workload represents a single snippet and therefore consists of the data of less than 32 samples (the exact number depends on the amount of outliers removed, but is never less than 16). This is far lower than recommended and commonly used sample sizes in ERP research, and therefore these statistics are of limited significance without additional data to confirm or refute them.

4.3.5 Ecological Validity

Many important factors that contribute to mental workload in real world code comprehension, such as loops or conditionals are currently not supported by the ACT-R model of code comprehension we used, and as such were not available for our snippet designs. Therefore, the generalisability of our findings to more complicated snippets or real-world examples of code comprehension is sharply limited. Future work extending the ACT-R model to be able to process more diverse snippets could enable us to simulate snippets that are significantly closer to the real world.

An additional issue that needs to be considered when the ACT-R model is extended is that there is no guarantee that the workload prediction algorithm we used is applicable to more complex settings with more causes of mental workload. While, on a technical level, it can easily be applied to even a far more complicated ACT-R model, we cannot guarantee that it would actually cover new causes of workload that such a model is intended to simulate. We have already seen an example of such an issue in our study, with the model apparently being unable to account for workload induced by calculations. It is important to note here that such an issue could both be caused by the workload prediction algorithm being unable to account for a source of mental workload, or by the underlying ACT-R model treating that source wrongly. As we outlined in our discussion, we suspect that, for the calculation difficulty in our study the latter is the case. However, an additional study with an ACT-R model that incorporates a better handling of calculations is necessary to clearly identify if the underlying model is indeed the issue here.

5

Related Work

In this thesis, we used a cognitive architecture to predict mental workload in code comprehension tasks. To our knowledge, no other study at the intersection of these three areas exists. In this chapter, we instead focus on research combining two of these areas. We present works measuring mental workload in code comprehension, applying cognitive architectures to code comprehension, and predicting mental workload with cognitive architectures.

5.1 Mental Workload and Code Comprehension

An attempt at constructing a model to predict the difficulty of a code comprehension task based on measurements of mental workload was made by Fritz et al. [24] in 2014. They combined physiological measurements from eye-tracking, EEG, and electrodermal activity in their model. Electrodermal activity measures the conductance of the skin and can be used to measure attention, stress, and emotional states. They built Naive Bayes Classifiers with features from all three sensors individually and all combinations of sensors. For predicting if a new participant would find a task easy or difficult, they found that the best classifier combined eye-tracking and electrodermal activity sensors, for a precision of 70.46% and a recall of 62.20%

Kosti et al. [44] built a model that used exclusively EEG data to predict subjective difficulty. Their model used Phase Locking Values, which quantify how strongly two electrodes synchronize in a specific frequency band, as its features. Notably, increased synchronization does not necessarily align with the measures of spectral band power used in this thesis, as two electrodes can have low power in a band but still oscillate synchronously. They built an Ordinal Regression Model with additional feature selection to obtain an optimal model. All frequency bands were present in the final model in at least two features, and there were no clear trends towards a positive or negative impact of increased synchronization in a specific band on subjective workload. Over four levels of reported difficulty they reported 55% accuracy and a mean absolute error of 0.51.

Medeiros et al. [53] built a model using relative spectral band power, among other features. They trained a Linear Discriminant Analysis Classifier on a set of features extracted from EEG data, to predict, for a given data recording, to which of three snippets it belongs. Before training, the feature set was reduced to a subset of the most important features using the ReliefF filter method. They evaluated their classifier in a leave-one-out cross validation procedure, in which they found that they achieved an accuracy of 70%. A more detailed analysis

revealed that the classifier could reliably differ between the easy snippet and the other two snippets but could not reliably differ between the medium and high complexity snippets. This aligned with the results for subjective difficulty they obtained from the NASA-TLX questionnaire, in which the easy snippet had significantly lower subjective workload than the other two snippets, which showed no significant difference in subjective workload. An analysis of the features used in the classifier showed that most of the important features were relative spectral band power measurements. While all frequency bands were present in the features, the most common bands in the classifier of code complexity were the gamma and theta bands. Although the generalisability of their results is lacking due to the limited set of snippets, Medeiros et al. showed that EEG data can be used to differentiate between snippets if the subjective workload during code comprehension shows a large enough difference.

An alternative approach to predicting subjective workload was employed by Alakmeh et al. [3], who trained a deep neural network on eye tracking data and program code to predict the subjective code difficulty and the correctness of a participant. They interpreted their subjective workload data as binary data, with all snippets below the mean being classified as low difficulty, and all snippets above the mean as high difficulty. Overall, they found their model to be well suited to predicting subjective difficulty, achieving an area under curve (AUC) score of 74%, both for predicting the subjective difficulty of a new participant on a known snippet and of a known participant on a new snippet.

Recently a metric of code complexity based on eye-tracking data was proposed by Hao et al. [31]. They used the number of revisits to a code area as a metric of code complexity, using eye-tracking to record how often a programmer looks at specific regions in the code, with more revisits to a region indicating a higher mental workload when working on that code region. They found that their metric aligned significantly better with EEG-based measurements of mental workload than traditional code complexity metrics do and highlighted that, unlike code complexity metrics this new metric is individual per programmer, taking into account individual abilities. It is, however, important to note that this comes with a usability caveat, as this metric requires a programmer to actively work on the code before it can give a value. As EEG-based measurements of workload, they used the ratio of theta band power to the combined alpha and beta band power at F2 and the ratio of theta band power to alpha band power at Pz.

The integration of physiological code complexity measurements into Integrated Development Environment (IDE)s was investigated by Stolp et al. [75]. They tested their tool CognitIDE in a feasibility study with 4 participants. CognitIDE allows for the incorporation of physiological data, such as EEG and eye-tracking data, into the IDE, and supports the simultaneous recording of physiological and behavioural data while participants are using the IDE to complete tasks. CognitIDE provides highlighting for areas of the code programmers look at the most, thereby marking areas that cause high mental load. In their feasibility study, Strolp et al. found that participants generally rated CognitIDE's highlighting to be accurate, and when investigating EEG data recorded in the study found that the synchronization in alpha and theta bands changed when engaged in work tasks, as

opposed to baseline recordings. This work by Strolp et al. highlights a possible future way to include physiological measurements of workload in development processes.

5.2 Cognitive Architectures and Mental Workload

An early exploration of predicting mental workload using a cognitive architecture was conducted by Lebiere et al. [46]. They simulated a simplified air traffic control task in ACT-R and compared the simulated data with empirical data collected from a study with a total of 48 participants. The tasks used in the study differed based on if they used a visual or text interface, and how fast participants had to respond to stimuli. They found that the ACT-R simulation matched well with the empirical data both in the time needed for a task and the errors made in the task. For the workload prediction they used the ratio of time spent in critical unit tasks to the total time in a task. They found that this gave a good estimate of participants' self-evaluation of workload for the different task categories.

While the workload prediction based on time spent in critical unit tasks worked well, it has two major weaknesses. The first weakness is that it is highly specific to a task, requiring the modeller to define what tasks count as critical. Secondly, it assigns the same value to each critical task, even though it is clearly possible for tasks to meaningfully contribute to workload, but have different workloads associated with them. In their work, Jo et al. [39] proposed a new workload prediction algorithm, using the activation times of ACT-R modules as the basic measurement of workload, with each module being assigned a different weight to represent different types of activity contributing differently to overall workload. The resulting workload prediction algorithm is the one we used in our study. In their validation of the workload prediction, Jo et al. used three different tasks. A memory task, in which participants had to recall a number associated to a word shown to them from a list of word-number pairs; a visual-manual task, in which participants had to click a number of buttons appearing on screen; and a menu selection task, in which participants had to select the value of a calculation from a menu of numbers. Each task was presented in different versions to vary the difficulty, for example the memory task featured two different lengths of lists of pairs and either word-number pairs with pre-existing association (e.g. "three" - 3) or without such an association (e.g. "gwjt" - 4). They found a linear relationship of the predicted workload and the reported subjective workload in the NASA-TLX that held across all tasks and difficulty levels.

Building on this workload prediction algorithm, Park et al. [63] proposed an extension of the algorithm to allow for the prediction of the sub-scales of the NASA-TLX. To compute the individual sub-scales, they used only relevant modules for that specific sub-scale, applying to those modules the same basic calculation as Jo et al. For example, they used only the workload caused by the motor modules to calculate the physical demand sub-scale of the NASA-TLX. They also introduced two new components in the workload prediction: the ratio of time spent on a task to the overall time available, which they used in computing the temporal demand and performance sub-scales; and a speed-up factor, in the form of an increased processing rate under time pressure, to account for an increase in human

processing speed in situations of time pressure. To evaluate their workload prediction, they used a simulated air traffic control scenario with 4 different conditions and found that all six sub-scales of the NASA-TLX correlated strongly with the relevant prediction and showed a linear relationship with the prediction.

Another extension of the algorithm by Jo et al. was proposed by Oh et al. [61], who integrated a mechanism to account for the workload of prior subtasks in the model. In their version of the model, the workload at a given time was calculated as the workload caused by the current task, plus the sum of the workloads of all prior subtasks, with the contribution of earlier tasks decaying based on the time elapsed since the tasks were finished. This workload prediction was then used to predict task switching behaviour in a multi-task study. In the study, participants had to play a memory game as one task and count the number of different images in a second task, between which they could switch when they wanted to. When the workload prediction predicted a high workload for the current task, it caused the ACT-R model to switch tasks. The model overall could predict well when participants switched tasks, and also predicted the performance of human participants in settings with and without task switching.

An alternative approach to predicting mental workload in a cognitive architecture was explored by Cao et al. [13], who used the cognitive architecture Queueing Network - Adaptive Control of Thought Rational (QN-ACTR), combining the ACT-R cognitive architecture with the Queueing Network (QN) cognitive architecture. The QN cognitive architecture simulates human cognition as a group of servers representing different capacities of the human mind, with each server having an input queue that it processes sequentially. The main advantage of the queueing network cognitive architecture is that it allows for a built-in simulation of multi-tasking behaviour, whereas, for example, ACT-R requires the modeller to specify the multi-tasking behaviour in the rules of the model for each task. The idea behind QN-ACTR is to simulate each of the ACT-R modules as its own server, with input queues replacing the buffers of the ACT-R system. To predict the workload in this system, Cao et al. proposed the Overall Expected Utilization (OEU) of the network. OEU is calculated as the ratio of the time required for all tasks given to the model to the time available to the model. The required time is here calculated based on an optimal setting, in which the model always has enough time for a task. The predicted workload of the model was compared to subjective workload measures from two pre existing experiments. A single-task study by Colle and Reid [20], where participants had to judge if two words shared a semantic category (e.g. "dog" and "cat" are both in the category "animal") and difficulty was modulated by the speed at which word pairs appeared and a multi-task study by Krerick and Allender [42], where participants had to complete a target shooting simulation and solve addition problems. Here, difficulty was modulated by three variables: the time for which targets were available; if there were friendly targets that participants were not allowed to shoot; and if the addition tasks were included in the study or not. For both studies, the OEU-based workload prediction showed very strong correlation and a linear relationship with the subjective workload ratings recorded in the study.

5.3 Cognitive Architectures and Code Comprehension

Cognitive models in the domain of programming were, for a long time, primarily focused on modelling program generation, i.e. the writing of new code. While these models give important insight into code comprehension, they do not model it directly. One example of such a cognitive architecture is the ACT-R Programming Tutor (APT) by Corbett [21], which was first presented in 2000. APT is built around a model based on the cognitive architecture ACT*, a predecessor of ACT-R. The goal of APT was to aid students in learning LISP by providing them with exercises fitting to areas where they struggle and by helping them during exercises by giving them hints when necessary. The ACT* model contained rules for solving programming problems in LISP. These rules are used by the tutor to trace a student's path to a solution, and to trace a student's knowledge over multiple tasks. When solving a problem, the student selects a sub-goal (typically a stand-in for an expression) and selects an action to solve that goal (e.g. choose a fitting expression). If the action matches one of the production rules in the ACT* model that match the current state, it is accepted and the state is updated; otherwise, it is rejected, and the student has to try again. The model traces whether a student knows a given production rule, based on if they select that production rule when given the chance, and additionally learns parameters on each individual student to match the learning prediction to their skill level. These parameters include how likely the student is to already know a rule, how likely they are to learn it after using, how likely they correctly guess an unlearned rule, and how likely they are to make a mistake despite knowing a rule. For each rule, the probability of the student having learned it is displayed by the tutor, and training stops when the probability of each rule being learned exceeds a pre-defined threshold. Until the threshold is reached, the tutor will keep presenting new problems for the student to solve. This approach is called cognitive mastery learning. When evaluating cognitive mastery learning compared to learning with a fixed set of problems, Corbett found that both in lab studies and in a field study in a course that ran for one semester, students showed significantly better performance using cognitive mastery learning. In the final exam of a course in which half of the students used cognitive mastery learning and half got a fixed set of problems the students that did use cognitive mastery learning achieved an average score of 85% and the other half achieved an average score of 72%.

Chiarelli [16] constructed a model to simulate learning programming with the help of self-explanation. Self-explanation refers to a learner formulating their own explanation of learned text. The model was implemented in the Python ACT-R cognitive architecture, a version of ACT-R that is implemented in Python instead of LISP. It is not an exact implementation of the LISP based ACT-R architecture in Python, but is an alternative model based on the same theory. The model is presented with instructional text, and self-explains each line. The model could simulate expert and novice students, with its initial knowledge in declarative memory, and the type of self-explanation conducted depending on the type of student. Overall, the model aligned with expectations, with all simulated students learning new knowledge from self-explanation and the used modes of self-explanation aligning with results from other domains. Notably, so far this model has only been compared to pre-existing data, mostly from other domains, and no study has been conducted to judge

how well it aligns with actual learning of programming skill.

Vorobeva [79] constructed Python ACT-R models to model different problem-solving techniques for programming and tested which of the strategies align best with problem solving approaches by novices and experts. They constructed five different models, each implementing a different problem-solving technique. These techniques were found in talk-aloud interviews with novices and experts beforehand. The models were not generalized problem solving models, but specialized for the specific task that was implemented. They found that some novices were best represented by a model that used a plan focus, a core step that is written initially, with the rest of the program being constructed to support that step. Experts and more competent novices were best modelled by a model using SGOMS, an extension of Python ACT-R, with planning units and tasks to allow for the modelling of more complex behaviour. Each planning unit consists of multiple tasks, that all have to be finished for the model to finish the planning unit. Planning units can be interconnected by a task in one planning unit calling for the beginning of another. In total, this creates a hierarchy of tasks that the model follows.

As described earlier, Closheim [18] developed the model of code comprehension that we used in our study. This model is able to read code in a limited subset of Python and calculate what the correct output for a code snippet is. Notably, this model works regardless of the specific snippet, so long as the snippet is contained to the subset of Python the model can process. They fine-tuned the model based on data obtained in a study with human participants on the same snippets, and found that ACT-R was able to match the answering time of human participants very well. Regarding the correctness of answers, they found that the model, while still performing acceptably, had issues with modelling different types of errors. Overall, Closheim showed that it is possible to construct a cognitive model of code comprehension, and that ACT-R is well suited to do so.

6 Concluding Remarks

In this section we summarise the findings of our study and outline possible directions for future work.

6.1 Conclusion

While identifying complex code early is highly beneficial in software engineering, the methods currently used to do so suffer from significant flaws. Code complexity metrics have well-known shortcomings in actually predicting code complexity, while neuro-imaging based approaches are difficult to use and do not scale. Cognitive architectures such as ACT-R could resolve this issue because they can combine advantages of code complexity metrics and neuro-imaging methods. An ACT-R model can incorporate findings from neuro-imaging methods during development and the developed model can later be applied to code without requiring additional measurements.

In our study, we showed that an existing ACT-R model of code comprehension can be applied to a new set of code snippets and, with fine-tuned parameters, still predict behavioural data. In addition, we showed that it is possible to use this ACT-R model to predict subjective workload in simple code comprehension tasks.

We investigated ERPs relative to a new line being shown in a code comprehension task, and found that the theorised inverse relationship between P300 amplitude and mental workload did not appear. However, we did find a late positive wave that grows with higher calculation length and outlined possible identities of this wave based on existing ERP research.

We conducted a first analysis of the relation of ACT-R workload prediction and EEG-based workload measures, and used the results of that analysis to identify areas in which the ACT-R model can be further improved. We found that the model currently does not represent some sources of mental workload well. This result of our analysis shows the importance of using workload measurements in future developments of the ACT-R model, as it shows that these measurements can be a valuable tool in identifying areas of a model that need to be improved.

In total, our findings support the feasibility of using an ACT-R model to simulate code comprehension, and the use of that simulation to predict programmers' mental workload during code comprehension. Our study also highlights some important limitations and pos-

sible areas of improvement for the ACT-R model, such as its lacking handling of calculations and redefined variables. It is crucial for future research to extend and improve the existing model so that it can simulate human behaviour and cognitive processes as accurately as possible.

6.2 Future Work

In our analysis of the predicted workload in the ACT-R model, we identified some interesting avenues for future work in the underlying area of researching mental workload in program comprehension, independently of ACT-R:

- **Theta-to-alpha ratio and mental workload**: In the past, the theta-to-alpha ratio has been used as a general measure of mental workload in code comprehension settings. We found that, while it is a good representation of working memory load, it does not measure the difficulty of calculations well. Future work shall examine how well the theta-to-alpha ratio captures different sources of mental workload in program comprehension, as it could be a crucial issue for studies using the theta-to-alpha ratio if it misses a central aspect of workload in program comprehension.
- **Details of the ERP Component**: Future studies shall further analyse the ERP component we found, to investigate in detail how it changes with changing calculation size, calculation difficulty, and working memory load. It is important that such work ensures that these components are varied independently from each other to allow clearly identifying the causes of this component. Additionally, future work shall investigate which of the four possible identities we proposed for this component matches with its characteristics. This would allow us to draw connections between code comprehension and other fields of ERP research.

Based on our findings regarding the performance of the ACT-R model and what causes errors in the time, correctness, and workload predictions, we propose the following future changes to the ACT-R model to allow it to better model program comprehension on the snippets we used in our study:

- **Repetition of current values**: To enable the model to better discern between current and old values of variables, it could be helpful to implement a strategy where the model sub-vocalizes the current values of variables. This aligns with the strategy used by most of our human participants and would increase the activation of the currently correct chunk for a variable.
- **Better representation of calculations**: Based on the problem size effect found in mental arithmetic studies, it would likely be appropriate to reduce the activation value of larger calculations, and thereby make the problem take longer to solve them. Additionally, it might make sense to implement a counting strategy the model can use to solve math problems it cannot recall, to match its behaviour to theories of mental arithmetic.
- **Results in memory**: Storing the preliminary results of longer calculations in declarative memory instead of the goal buffer would slow down the model in snippets with

longer calculations by requiring an additional recall to access the preliminary result after accessing the next value used in a calculation, thereby increasing the expected workload in such cases to align closer to reality. Alternatively, we could use the imaginal module to store the preliminary results. This would prevent failed recalls of the preliminary results, but would still increase the time needed to access them, as the imaginal module, in contrast to the goal module, does not act instantaneously.

- **Calculation errors**: By making it possible for the model to miscalculate a result, more types of errors could be simulated, and therefore the model could better match human performance. For this, similarity matrices and spreading activation between the calculations could be used to represent likely errors in calculations.

Additionally, to this work on the model for the current task it is used on, future work shall extend the model to be used for more complicated program comprehension tasks. For this, it is necessary to extend the capabilities of the model in handling code. The first step in this would likely be to extend the model to be capable of a natural reading flow when presented with entire snippets, instead of the current line-by-line presentation. Possible further extensions could include the model being able to handle additional data types, the implementation of lists in the model or the implementation of a more complex control flow in the form of conditionals, loops, and function calls. Once such an extended model exists, the workload prediction method we described could be used to predict the code complexity of much more complex and varied code. This would represent an important step towards the prediction of the complexity of arbitrary code in a practical development setting.

A

Code Snippets

```
x = 6
m = 4
c = 2
k = 8
a = x * c - k
print(a)
```

Snippet 1

```
x = 3
a = 7
x = 6
a = a + x
print(a)
```

Snippet 2

```
k = 4
c = 9
c = 3
p = 6
k = 7
p = 5
m = k + c
k = p - c
k = m * k
print(k)
```

Snippet 3

```
x = 9
v = 6
c = 5
x = 3
p = 4
a = p + x
m = x * 2 -p + a - c
print(m)
```

Snippet 4

```
b = 6
n = 4
g = 5
b = 2
z = g - n + b
print(z)
```

Snippet 5

```
j = 2
u = 3 * 2
q = 5 - j
w = u + 7
j = 6 - 4
s = w + j - q
print(s)
```

Snippet 6

```
v = 6
x = 4
x = 3
m = x + v
p = 1
x = 4 - p
v = x + m
print(v)
```

Snippet 7

```
r = 2
n = r * 8
print(n)
```

Snippet 8

```
m = 4
a = 7 + 2
x = a / 3
m = 5
c = 3 + 1
p = a - c + x
print(p)
```

Snippet 9

```
x = 4
k = 7
p = 8
v = 5
k = 2
v = p - x + k
print(x)
```

Snippet 10

```
c = 8
m = 2
x = c / m
print(x)
```

Snippet 11

```
n = 5
r = 3
r = 7
n = r - 3
b = 3
n = n * b + 5 - r + n
print(n)
```

Snippet 12

```
g = 4
n = 6
r = 3
h = 5
b = 1
z = b * 4 - g
print(z)
```

Snippet 13

```
u = 5
j = u - 4
f = 1
f = 2
e = j * f
print(e)
```

Snippet 14

```
m = 5
m = 8
v = 4
v = m - 2
m = v / 2
p = 5
m = m + p
v = m - 4
m = v * 2
v = 2
p = m / v
print(p)
```

Snippet 15

```
c = 5
c = 2
k = 8
k = k * c
print(k)
```

Snippet 16

```
k = 3
x = 2
c = k * x
print(c)
```

Snippet 17

```
a = 9
k = 4
k = 5
a = 7
v = 3
m = a + k
k = m / v
print(k)
```

Snippet 18

```
x =  7
c = x - 4
print(c)
```

Snippet 19

```
h = 4
b = 8
n = b * 2
z = h * 4 - n
print(z)
```

Snippet 20

```
w = 2
u = 8
j = u / 4
w = 5
f = j * w
print(f)
```

Snippet 21

```
f = 6
u = 3
f = f * u
print(f)
```

Snippet 22

```
s = 8
u = s / 2
w = u + 6
f = w - s
s = f * 6 - w + s
print(s)
```

Snippet 23

```
s = 1
e = 4
u = 3
s = 5
q = s - e
u = 4
s = q + u
e = s * 2 + e
print(e)
```

Snippet 24

```
b = 5
b = 3
h = 2
z = 2
n = h * z
y = n * 4
r = y / z
g = b * 2
print(r)
```

Snippet 25

```
u = 7
s = 8
q = u - 3
u = s / q
s = q / 2 * 3 + 5 + u
print(s)
```

Snippet 26

```
k = 5
v = 3
k = 10
c = 4
x = 14 - k
m = v * 4 + c + x
print(m)
```

Snippet 27

```
p = 5
m = 2
p = 4
x = 8
v = p * m - x
print(v)
```

Snippet 28

```
s = 5
q = 9
w = 4
e = q - s
print(e)
```

Snippet 29

```
s = 3
j = 6
w = 5
q = s + j - w
print(q)
```

Snippet 30

B

Postquestionnaire

Category	Count
Total	34
Male	23
Female	11
Finished Bachelor	25
no degree yet	7
Finished Master	2
Student with CS-Job	7
Student	25
PhD student	2
Programming Experience	3.765 ± 0.431
Logical Programming	2.412 ± 1.328
Functional Programming	2.412 ± 1.234
Imperative Programming	3.382 ± 0.853
Object-Oriented Programming	2.647 ± 0.981

Figure B.1: Demographic data from our study, categorical values are given as the count of answers, experience values as mean and standard deviation

1. Pariticpant-ID

2. Could you complete the tasks without distractions?

- ◯ Yes
- ◯ No

3. What gender do you identify as?

- ◯ male
- ◯ female
- ◯ trans male
- ◯ trans female
- ◯ non-binary
- ◯ Other:
- ◯ Prefer not to answer

4. What is your age (in years)?

5. What country do you currently live in?

- ◯ Germany
- ◯ Other:

6. What is your formal education in computer science, software engineering?

- ◯ No formal qualification, self- taught
- ◯ Vocational training program, course
- ◯ No degree yet, studied Computer Science (or a related field) for
- ◯ University degree (undergraduate, bachelor level)
- ◯ University degree (graduate, master level)
- ◯ University degree (graduate, PhD level)
- ◯ Other:

Figure B.2: The first page of the post-questionnaire used in our study.

7. What is your profession?

8. How would you rate your programming experience?

	very inexperienced	inexperienced	neutral	experienced	very experienced	do not want to answer
I am:	○	○	○	○	○	○

9. How experienced are you with the following programming paradigms?

	very inexperienced	inexperienced	mediocre	experienced	very experienced
logical (Prolog)	○	○	○	○	○
functional (Haskell)	○	○	○	○	○
imperative (C)	○	○	○	○	○
object-oriented (Java)	○	○	○	○	○

Last Page

Thank you for participating in this study!

Yannick Lehmen, Universität des Saarlandes – 2025

Figure B.3: The second page of the post-questionnaire used in our study. After this participants were verbally asked if they had a specific strategy they used for the code comprehension tasks.

C

Additional Correlations

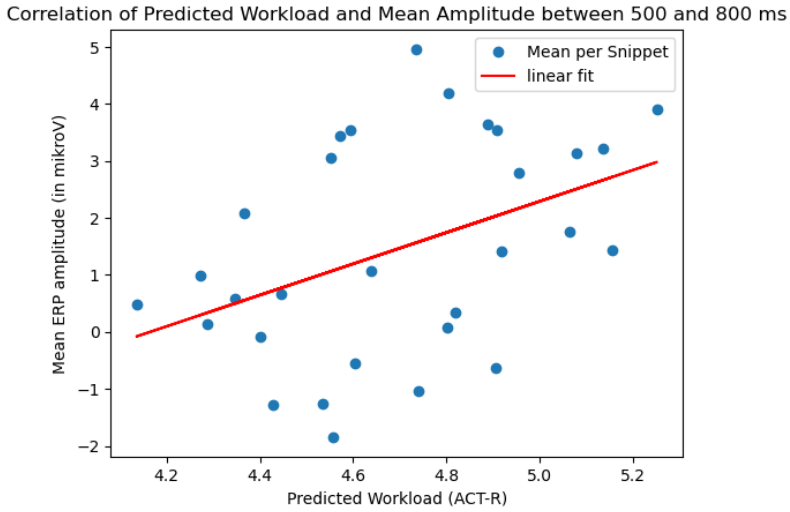

Figure C.1: Correlation of the late ERP wave and the predicted workload (Kendall's Tau = 0.287)

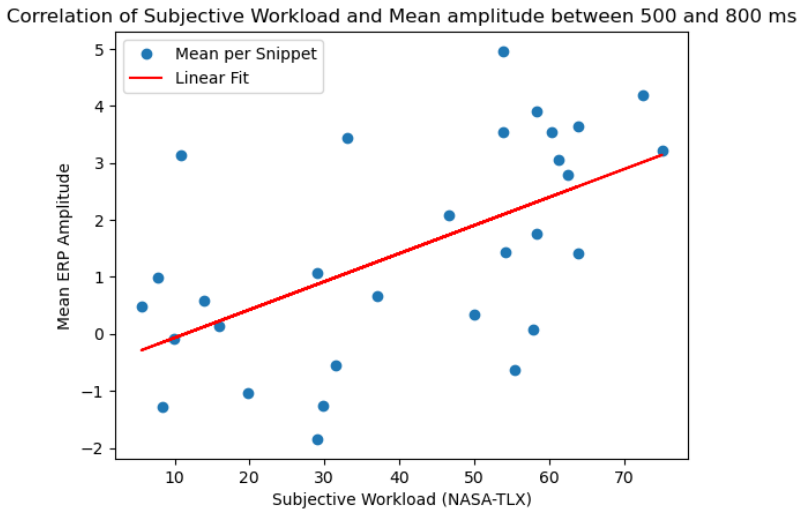

Figure C.2: Correlation of the late ERP wave and the subjective workload (Kendall's Tau = 0.398)

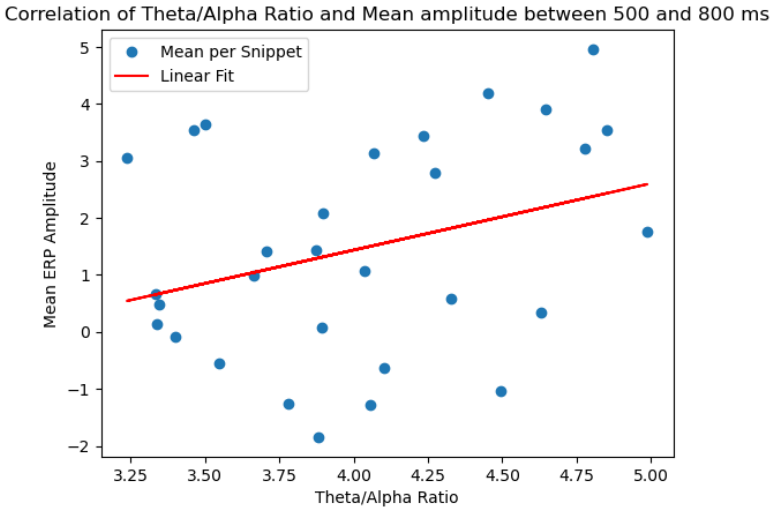

Figure C.3: Correlation of the late ERP wave and the theta-to-alpha ratio (Kendall's Tau = 0.205)

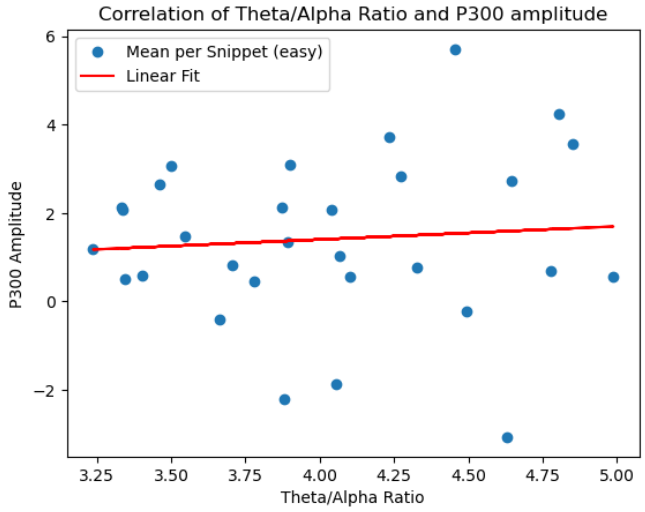

Figure C.4: Correlation of the P300 amplitude and the theta-to-alpha ratio (Kendall's Tau = 0.205)

D

Optimization of the ACT-R Model

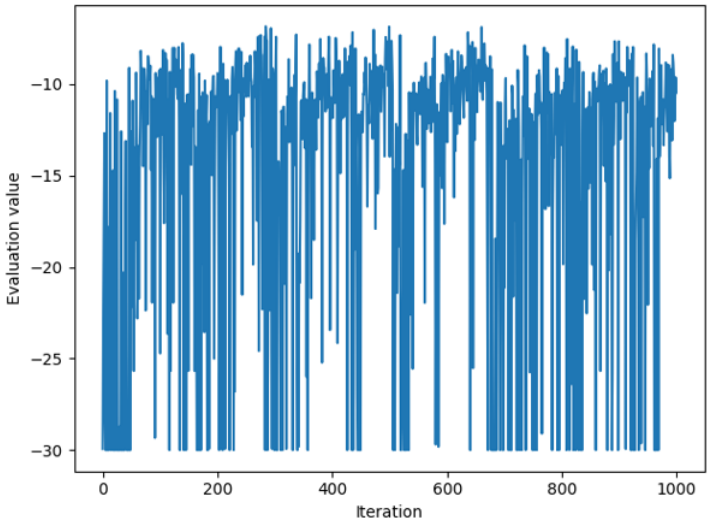

Figure D.1: Time course of the negated KS-Statistic during the optimization of the ACT-R model

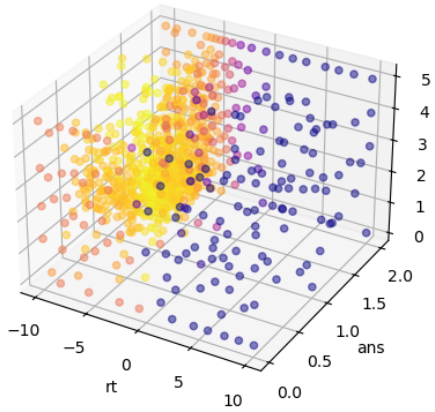

Figure D.2: Graphical representation of the optimization space, brigther colors represent better results. The z-axis is the latency factor.

Statement on the Usage of Generative Digital Assistants

For this thesis, the following generative digital assistants have been used: We have used DeepL Write[1] for correcting spelling and grammar. This was done by enabling the corrections only mode in the tool, so that it did not propose rewrites and then manually checking each proposed correction for its applicability. We are aware of the potential dangers of using these tools and have used them sensibly with caution and with critical thinking.

1 https://www.deepl.com/en/write

Bibliography

[1] JN Acharya and VJ Acharya. "Overview of EEG Montages and Principles of Localization." In: *Journal of Clinical Neurophysiology: Official Publication of the American Electroencephalographic Society* 36.5 (2019), pp. 325–329.

[2] Naser Al Madi, Siyuan Peng, and Tamsin Rogers. "Assessing workload perception in introductory computer science projects using NASA-TLX." In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 2022, pp. 668–674.

[3] Tarek Alakmeh, David Reich, Lena Jäger, and Thomas Fritz. "Predicting Code Comprehension: A Novel Approach to Align Human Gaze with Code using Deep Neural Networks." In: *Proc. ACM Softw. Eng.* 1.FSE (July 2024).

[4] John R Anderson. *Rules of the mind*. Lawrence Erlbaum Associates Inc., 1993.

[5] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglas, Christian Lebiere, and Yulin Qin. "An Integrated Theory of the Mind." In: *Psychological Review* 111.4 (2004), pp. 1036–1060.

[6] Rajendra K. Bandi, Vijay K. Vaishnavi, and Daniel E. Turk. "Predicting maintenance performance using object-oriented design complexity metrics." In: *IEEE transactions on Software Engineering* 29.1 (2003), pp. 77–87.

[7] Rajiv D Banker, Srikant M Datar, Chris F Kemerer, and Dani Zweig. "Software complexity and maintenance costs." In: *Communications of the ACM* 36.11 (1993), pp. 81–95.

[8] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing." In: *Journal of the Royal statistical society: series B (Methodological)* 57.1 (1995), pp. 289–300.

[9] Annabelle Bergum, Anna-Maria Maurer, Norman Peitek, Regine Bader, Axel Mecklinger, Vera Demberg, Janet Siegmund, and Sven Apel. "Unexpected but informative: What fixation-related potentials tell us about the processing of confusing program code." In: *arXiv preprint arXiv:2412.10099* (2024).

[10] Ruven Brooks. "Towards a theory of the comprehension of computer programs." In: *International journal of man-machine studies* 18.6 (1983), pp. 543–554.

[11] Anne-Marie Brouwer, Maarten A. Hogervorst, Jan B.F. van Erp, Tobias Heffelaar, PAtrick H. Zimmerman, and Robert Oostenveld. "Estimating workload using EEG spectral power and ERPs in the n-back task." In: *Journal of Neural Engineering* 9 (2012), pp. 14–28.

[12] James C Byers, AC Bittner, and Susan G Hill. "Traditional and raw task load index (TLX) correlations: Are paired comparisons necessary." In: *Advances in industrial ergonomics and safety* 1 (1989), pp. 481–485.

[13] Shi Cao and Yili Liu. "Modelling workload in cognitive and concurrent tasks with time stress using an integrated cognitive architecture." In: *International Journal of Human Factors Modelling and Simulation* 5.2 (2015), pp. 113–135.

[14] Mickael Causse, Eve Fabre, Louise Giraudet, Marine Gonzalez, and Vsevolod Peysakhovich. "EEG/ERP as a Measure of Mental Workload in a Simple Piloting Task." In: vol. 3. Dec. 2015, pp. 5230–5236.

[15] Jiayu Chen, Xinyi Song, and Zhenghang Lin. "Revealing the "Invisible Gorilla" in construction: Estimating construction safety through mental workload assessment." In: *Automation in Construction* 63 (2016), pp. 173–183.

[16] Veronica Chiarelli. "Learning basic python concepts via self-explanation: a preliminary python ACT-R model." In: *Proceedings of the 19th International Conference on Cognitive Modeling*. 2021.

[17] Moon Kyoung Choi, Seung Min Lee, Jun Su Ha, and Poong Hyun Seong. "Development of an EEG-based workload measurement method in nuclear power plants." In: *Annals of Nuclear Energy* 111 (2018), pp. 595–607.

[18] Christian Closheim. "Cognitive Modeling in Code Comprehension: An Empirical Study of Short-Term Memory Retrievals." Master's thesis. Saarbrücken, Germany: Saarland University, 2024.

[19] Andy Cockburn and Carl Gutwin. "Anchoring effects and troublesome asymmetric transfer in subjective ratings." In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–12.

[20] Herbert A Colle and Gary B Reid. "Context effects in subjective mental workload ratings." In: *Human factors* 40.4 (1998), pp. 591–600.

[21] Albert Corbett. *Cognitive Mastery Learning in the ACT Programming Tutor*. Tech. rep. AAAI, 2000.

[22] Hubert E Dunsmore. "Software metrics: an overview of an evolving methodology." In: *Information Processing & Management* 20.1-2 (1984), pp. 183–192.

[23] Timon Dörrzapf. "Advancing Cognitive Modeling of Code Comprehension: Integrating Backtracking and Similarity Error Mechanisms in ACT-R." Master's thesis, unreleased. Saarbrücken, Germany: Saarland University, 2025.

[24] Thoams Fritz, Andrew Begel, Sebastian C. Müller, Serap Yiglit-Elliott, and Manuela Züger. "Using psycho-physiological measures to assess task difficulty in software development." In: *Proceedings of the 36th International Conference on Software Engineering*. Association for Computing Machinery, 2014, pp. 402–413.

[25] Nancy R. Gee, Taylor Reed, April Whiting, Erika Friedmann, Donna Snellgrove, and Katherine A. Sloman. "Observing Live Fish Improves Perceptions of Mood, Relaxation and Anxiety, But Does Not Consistently Alter Heart Rate or Heart Rate Variability." In: *International Journal of Environmental Research and Public Health* 16 (2019).

[26] Dan Gopstein, Jake Iannacone, Yu Yan, Lois DeLong, Yanyan Zhuang, Martin Yeh, and Justin Cappos. "Understanding misunderstandings in source code." In: *2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017, pp. 129–139.

[27] Ana C. Gouvea, Colin Phillips, Nina Kazanina, and David Poeppel. "The linguistic processes underlying the P600." In: *Language and Cognitive Processes* 25.2 (2010), pp. 149–188. DOI: 10.1080/01690960902965951. URL: https://doi.org/10.1080/01690960902965951.

[28] ACT-R Research Group. *Reference Manual*. Website. Available online at http://actr.psy.cmu.edu/actr7.x/reference-manual.pdf; visited on January 23rd, 2025. 2025.

[29] ACT-R Research Group. *Tutorial Units*. Website. Available online at http://actr.psy.cmu.edu/actr7.x/units.zip; visited on January 23rd, 2025. 2025.

[30] Peter Hagoort, Colin Brown, and Jolanda Groothusen. "The syntactic positive shift (sps) as an erp measure of syntactic processing." In: *Language and Cognitive Processes* 8.4 (1993), pp. 439–483. DOI: 10.1080/01690969308407585. URL: https://doi.org/10.1080/01690969308407585.

[31] Gao Hao, Haytham Hijazi, Júlio Medeiros, João Durães, Chan Tong Lam, Paulo de Carvalho, and Henrique Madeira. "NRevisit: A Cognitive Behavioral Metric for Code Understandability Assessment." In: *arXiv preprint arXiv:2504.18345* (2025).

[32] David J Hardy and Matthew J Wright. "Assessing workload in neuropsychology: An illustration with the Tower of Hanoi test." In: *Journal of Clinical and Experimental Neuropsychology* 40.10 (2018), pp. 1022–1029.

[33] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research." In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183.

[34] Thomas Hinault and Patrick Lemaire. "What does EEG tell us about arithmetic strategies? A review." In: *International Journal of Psychophysiology* 106 (2016), pp. 115–126. ISSN: 0167-8760. DOI: https://doi.org/10.1016/j.ijpsycho.2016.05.006. URL: https://www.sciencedirect.com/science/article/pii/S0167876016300770.

[35] Anu Holm, Kristian Lukander, Jussi Korpela, Mikael Sallinen, and Kiti M. I. Müller. "Estimating brain load from the EEG." In: 9 (2009), pp. 639–651.

[36] Renske Huffmeijer, Marian J Bakermans-Kranenburg, Lenneke RA Alink, and Marinus H Van IJzendoorn. "Reliability of event-related potentials: The influence of number of trials and electrodes." In: *Physiology & behavior* 130 (2014), pp. 13–22.

[37] Yoshinobu Iguchi and Isao Hashimoto. "Sequential information processing during a mental arithmetic is reflected in the time course of event-related brain potentials." In: *Clinical Neurophysiology* 111.2 (2000), pp. 204–213. DOI: https://doi.org/10.1016/S1388-2457(99)00244-8. URL: https://www.sciencedirect.com/science/article/pii/S1388245799002448.

[38] Peter Jezzard, Paul M Matthews, Stephen M Smith, et al. *Functional MRI: an introduction to methods*. Vol. 61. Oxford university press Oxford, 2001.

[39] Seongsik Jo, Rohae Myung, and Daesub Yoon. "Quantitative prediction of mental workload with the ACT-R cognitive architecture." In: *International Journal of Industrial Ergonomics* 42.4 (2012), pp. 359–370.

[40] Frank J. Massey Jr. "The Kolmogorov-Smirnov Test for Goodness of Fit." In: *Journal of the American Statistical Association* 46.253 (1951), pp. 68–78.

[41] Otis M. Solomon Jr. *PSD computations using Welch's method. [Power Spectral Density (PSD)]*. Tech. rep. Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), Dec. 1991. URL: https://www.osti.gov/biblio/5688766.

[42] SE Kerick and LE Allender. "Effects of cognitive workload on decision accuracy, shooting performance, and cortical activity of soldiers." In: *Transformational Science And Technology For The Current And Future Force: (With CD-ROM)*. World Scientific, 2006, pp. 359–362.

[43] Timo Kirschstein and Rüdiger Köhling. "What is the Source of the EEG?" In: *Clinical EEG and Neuroscience* 40.3 (2009), pp. 146–149. DOI: 10.1177/155005940904000305.

[44] Makrina Viola Kosti, Kostas Georgiadis, Dimitrios A. Adamos, Nikos Laskaris, Diomidis Spinellis, and Lefteris Angelis. "Towards an affordable brain computer interface for the assessment of programmers' mental workload." In: *International Journal of Human-Computer Studies* 115 (2018), pp. 52–66. ISSN: 1071-5819.

[45] John E. Laird. *Introduction to Soar*. 2022. URL: https://arxiv.org/abs/2205.03854.

[46] Christian Lebiere, J. Anderson, and D. Bothell. "Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task." In: *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Jan. 2001.

[47] José León-Carrión and Umberto León-Domínguez. "Functional near-infrared spectroscopy (fNIRS): principles and neuroscientific applications." In: *Neuroimaging methods* 97 (2012), pp. 48–74.

[48] Luca Longo, Christopher D Wickens, Gabriella Hancock, and Peter A Hancock. "Human mental workload: A survey and a novel inclusive definition." In: *Frontiers in psychology* 13 (2022), p. 883321.

[49] Steven J. Luck. *An Introduction to the Event-Related Potential Technique*. The MIT Press, 2014.

[50] MNE-Tools. *MNE-Python*. Website. Available online at https://github.com/mne-tools/mne-python; visited on September 24th, 2025. 2025.

[51] Maurice H. Halstead. *Elements of Software Science*. Elsevier Science Inc., 1977.

[52] T.J. McCabe. "A Complexity Measure." In: *IEEE Transactions on Software Engineering* SE-2.4 (1976), pp. 308–320.

[53] J. Medeiros, R. Couceiro, J. Castelhano, M. Castelo Branco, G. Duarte, C. Duarte, J. Durães, H. Madeira, P. Carvalho, and C. Teixeira. "Software code complexity assessment using EEG features." In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019, pp. 1413–1416.

[54] Júlio Medeiros, Ricardo Couceiro, Gonçalo Duarte, João Durães, João Castelhano, Catarina Duarte, Miguel Castelo-Branco, Henrique Madeira, Paulo de Carvalho, and César Teixeira. "Can EEG Be Adopted as a Neuroscience Reference for Assessing Software Programmers' Cognitive Load?" In: *Sensors* 21.7 (2021).

[55] David E. Meyer, Jennifer M. Glass, Shane T. Mueller, Travis L. Seymour, and David E. Kieras and. "Executive-process interactive control: A unified computational theory for answering 20 questions (and more) about cognitive ageing." In: *European Journal of Cognitive Psychology* 13.1-2 (2001), pp. 123–164.

[56] Denise Neill. "Nursing workload and the changing health care environment: a review of the literature." In: *Administrative Issues Journal* 1.2 (2011), p. 13.

[57] Mayra Nilson, Vard Antinyan, and Lucas Gren. "Do internal software quality tools measure validated metrics?" In: *Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20*. Springer. 2019, pp. 637–648.

[58] M Isabel Núñez-Peña and M Luisa Honrubia-Serrano. "P600 related to rule violation in an arithmetic task." In: *Cognitive Brain Research* 18.2 (2004), pp. 130–141.

[59] M. Isabel Núñez-Peña, M. Luisa Honrubia-Serrano, and Carles Escera. "Problem size effect in additions and subtractions: an event-related potential study." In: *Neuroscience Letters* 373.1 (2004), pp. 21–25. ISSN: 0304-3940. DOI: https://doi.org/10.1016/j.neulet.2004.09.053. URL: https://www.sciencedirect.com/science/article/pii/S0304394004011991.

[60] Michael P O'Brien, Jim Buckley, and Teresa M Shaft. "Expectation-based, inference-based, and bottom-up software comprehension." In: *Journal of Software Maintenance and Evolution: Research and Practice* 16.6 (2004), pp. 427–447.

[61] Hyungseok Oh, Yongdeok Yun, and Rohae Myung. "Cognitive Modeling of Task Switching in Discretionary Multitasking Based on the ACT-R Cognitive Architecture." In: *Applied Sciences* 11.9 (2021).

[62] Fred GWC Paas and Jeroen JG Van Merriënboer. "The efficiency of instructional conditions: An approach to combine mental effort and performance measures." In: *Human factors* 35.4 (1993), pp. 737–743.

[63] Sungjin Park, Sungoo Jeong, and Rohae Myung. "Modeling of multiple sources of workload and time pressure effect with ACT-R." In: *International Journal of Industrial Ergonomics* 63 (2018), pp. 37–48.

[64] Aniruddh D. Patel, Edward Gibson, Jennifer Ratner, Mireille Besson, and Phillip J. Holcomb. "Processing Syntactic Relations in Language and Music: An Event-Related Potential Study." In: *Journal of Cognitive Neuroscience* 10.6 (Nov. 1998), pp. 717–733. DOI: 10.1162/089892998563121.

[65] Norman Peitek, Sven Apel, Chris Panin, André Brechmann, and Janet Siegmund. "Program Comprehension and Code Complexity Metrics: An fMRI Study." In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. ACM, 2021, pp. 524–536.

[66] Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. "Correlates of programmer efficacy and their link to experience: a combined EEG and eye-tracking study." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, pp. 120–131.

[67] Norman Peitek, Janet Siegmund, Chris Parnin, Sven Apel, Johannes C. Hofmeister, and André Brechmann. "Simultaneous measurement of program comprehension with fMRI and eye tracking: a case study." In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '18. Oulu, Finland: Association for Computing Machinery, 2018. ISBN: 9781450358231. DOI: 10.1145/3239235.3240495. URL: https://doi.org/10.1145/3239235.3240495.

[68] Weiwei Peng. "EEG preprocessing and denoising." In: *EEG Signal Processing and Feature Extraction*. Springer, 2019, pp. 71–87.

[69] F. Perrin, J. Pernier, O. Bertrand, and J.F. Echallier. "Spherical splines for scalp potential and current density mapping." In: *Electroencephalography and Clinical Neurophysiology* 72.2 (1989), pp. 184–187. ISSN: 0013-4694. DOI: https://doi.org/10.1016/0013-4694(89)90180-6. URL: https://www.sciencedirect.com/science/article/pii/0013469489901806.

[70] Susana Rubio, Eva Díaz, Jesús Martín, and José M. Puente. "Evaluation of Subjective Mental Workload: A Comparison of SWAT, NASA-TLX, and Workload Profile Methods." In: *Applied Psychology* 53.1 (2004), pp. 61–86. DOI: https://doi.org/10.1111/j.1464-0597.2004.00161.x.

[71] Simone Scalabrino, Gabriele Bavota, Christopher Vendome, Mario Linares-Vásquez, Denys Poshyvanyk, and Rocco Oliveto. "Automatically Assessing Code Understandability." In: *IEEE Transactions on Software Engineering* 47.3 (2021), pp. 595–613.

[72] Christian Scharinger, Alexander Soutschek, Torsten Schubert, and Peter Gerjets. "Comparison of the Working Memory Load in N-Back and Working Memory Span Tasks by Means of EEG Frequency Band Power and P300 Amplitude." In: *Frontiers in Human Neuroscience* 11 (Jan. 2017).

[73] Ben Shneiderman and Richard Mayer. "Syntactic/semantic interactions in programmer behavior: A model and experimental results." In: *International Journal of Computer & Information Sciences* 8 (1979), pp. 219–238.

[74] Janet Siegmund. "Program Comprehension: Past, Present, and Future." In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. Vol. 5. 2016, pp. 13–20.

[75] Fabian Stolp, Charlotte Brandebusemeyer, Franziska Hradilak, Lara Kursawe, Magnus Menger, Franz Sauerwald, and Bert Arnrich. *Using CognitIDE to Capture Developers' Cognitive Load via Physiological Activity During Everyday Software Development Tasks*. 2025. arXiv: 2503.03537 [cs.SE]. URL: https://arxiv.org/abs/2503.03537.

[76] Dianne E Thornhill and Cyma Van Petten. "Lexical versus conceptual anticipation during sentence processing: Frontal positivity and N400 ERP components." In: *International journal of psychophysiology* 83.3 (2012), pp. 382–392.

[77] Rebecca Tiarks. "What maintenance programmers really do: An observational study." In: *Softwaretechnik-Trends Band 31, Heft 2*. Geselllschaft für Informatik eV. 2011.

[78] Anneliese Von Mayrhauser, A Marie Vans, and Adele E Howe. "Program understanding behaviour during enhancement of large-scale software." In: *Journal of Software Maintenance: Research and Practice* 9.5 (1997), pp. 299–327.

[79] Maria Vorobeva. "Modelling Programming Problem Solving in Python ACT-R." Master's thesis. Ottawa, Canada: Carleton University, 2021.

[80] Christopher D Wickens, Sallie E Gordon, and Yili Liu. *An Introduction to Human Factors Engineering*. Addison Wesley Longman, 1998.

[81] Glenn F Wilson and Thomas F Eggemeier. "Mental workload measurement." In: *International encyclopedia of ergonomics and human factors* 1 (2006), p. 9.

[82] Marvin Wyrich. *Source Code Comprehension: A Contemporary Definition and Conceptual Model for Empirical Investigation*. 2023. arXiv: 2310.11301 [cs.SE]. URL: https://arxiv.org/abs/2310.11301.

[83] Marvin Wyrich, Justus Bogner, and Stefan Wagner. "40 Years of Designing Code Comprehension Experiments: A Systematic Mapping Study." In: *ACM Comput. Surv.* 56.4 (Nov. 2023). ISSN: 0360-0300. URL: https://doi.org/10.1145/3626522.

[84] N Jane Zbrodoff and Gordon D Logan. "What everyone finds: The problem-size effect." In: *The handbook of mathematical cognition*. Psychology Press, 2005, pp. 331–345.