Identifying program confusion using electroencephalogram measurements

Martin K.-C. Yeh, Yu Yan, Yanyan Zhuang & Lois Anne DeLong

To cite this article: Martin K.-C. Yeh, Yu Yan, Yanyan Zhuang & Lois Anne DeLong (2021): Identifying program confusion using electroencephalogram measurements, Behaviour & Information Technology, DOI: 10.1080/0144929X.2021.1933182

To link to this article: https://doi.org/10.1080/0144929X.2021.1933182

Published online: 05 Jun 2021.

BEHAVIOUR & INFORMATION TECHNOLOGY
https://doi.org/10.1080/0144929X.2021.1933182

Identifying program confusion using electroencephalogram measurements

Martin K.-C. Yeh[a], Yu Yan[b], Yanyan Zhuang[c] and Lois Anne DeLong[d]

[a]Information Sciences and Technology, Pennsylvania State University-Brandywine, Media, PA, USA; [b]Teaching + Learning Commons, University of California, San Diego, CA, USA; [c]Computer Science, University of Colorado, Colorado Springs, CO, USA; [d]Tandon School of Engineering, New York University, Brooklyn, NY, USA

ABSTRACT
In this paper, we present an experimental study in which an electroencephalogram (EEG) device was used to measure cognitive load in programmers as they attempted to predict the output of C code snippets. Our goal was to see if particular patterns within the snippet induced higher levels of cognitive load, and if the collected EEG data might provide more detailed insights than performance measures. Our results suggest that while cognitive load can be an influence on code comprehension performance, other human factors, such as a tendency to forget certain programming rules or to misread what the code is asking them to do may also play a role, particularly for novice programmers. We conclude that: (1) different types of code patterns can affect programmers' cognitive processes in disparate ways, (2) neither self-reported data nor brainwave activity alone is a reliable indicator of programmers' level of comprehension for all types of code snippets, (3) EEG techniques could be useful to better understand the relationships between program comprehension, code patterns and cognitive processes, and (4) tests like ours could be useful to identify crucial learning gaps in novice programmers, which, in turn can be leveraged to improve programming tools and teaching strategies.

ARTICLE HISTORY
Received 12 January 2021
Accepted 13 May 2021

KEYWORDS
Electroencephalogram; EEG; brainwave; cognitive load; program comprehension

1. Introduction

Every day new computer programs are created by programmers of varying skill levels, using programming languages that also vary in levels of complexity. Computer programs use source code that contains various symbols, operators and statements to control the execution of applications. To a computer, code is unambiguous, precise and can be interpreted only one way. To a human programmer, however, the same code can potentially be interpreted in many different ways. Just as misunderstandings about words and phrases can occur in any natural language, due to idiomatic expressions or unfamiliar syntactic structures, programmers can also misread the placement of various constructs within code. To make matters worse, such misunderstandings can happen even when programmers deal with relatively short code snippets.

Our recent study (Gopstein et al. 2017) identified a few patterns in the C language that can cause programmers to be confused as to how a given piece of code might behave. In this context, we define *confusion* as when a programmer – presented with a deterministic, syntactically and semantically valid piece of code – believes a code snippet would output something different from what a computer would actually execute. The minimal portions of code that can induce this confusion are named *atoms of confusion* or *atoms* for short. We tested the impact of these code patterns by asking participants to predict outputs for snippets, half of which contained an atom. The other half contained the same code, but the atom was removed and replaced by functionally equivalent, but less confusing statements, a process called a *transformation*. Results from the study affirm that atoms could lead to a significantly increased rate of misunderstanding as measured in performance scores. Such misunderstandings can complicate the already difficult task of program comprehension – a software development activity that can take up to 50%–90% of program maintenance efforts (Lientz, Swanson, and Tompkins 1978).

The link between the presence of atoms and bugs was confirmed in a follow-up study (Gopstein et al. 2018), which reported that atoms were prevalent in many large, popular open-source projects, including git, clang and mysql-server. Additionally, we found that both the frequency of bug fixes and comments were correlated with the presence of atoms. As software programmers develop more complex applications, the number of instances where misunderstandings can lead to bugs will likely increase as well. Thus knowing

CONTACT Martin K.-C. Yeh martin.yeh@psu.edu 207H Tomezsko Building, Penn State Brandywine, 25 Yearsley Mill Rd, Media, PA 19063, USA
This article has been republished with minor changes. These changes do not impact the academic content of the article.

more about the role of these atoms in creating misunderstandings could help programmers avoid behaviours that could have risky consequences.

Unfortunately, the most commonly used methods for assessing code comprehension are not effective for exploring sources of confusion. One approach is to have programmers complete a series of tasks and evaluate comprehension using some type of performance indicators, such as completeness or correctness of answers (Corritore and Wiedenbeck 1991; Teasley 1994; McKeithen et al. 1981; Romero 2004). However, this method offers little insight into where and why misunderstandings might occur. Furthermore, the value of performance measures as an indicator of confusion is questionable, as other unrelated variables, such as typographic errors or the participants' years of programming experience could also affect these measures. A second approach, Verbal Protocol Analysis (VPA) or the "think-aloud" method (Ericsson and Simon 1984), also has its limitations. This strategy asks participants to talk aloud as they perform a programming task. Hints to the participants' thought processes are then gleaned from their comments. Unfortunately, VPA is not effective if participants lack the necessary verbal skills to explain their work process (Adelson and Soloway 1985; Letovsky 1987; Pennington 1987a; Wilson 1994; Atman and Bursic 1998; Yeh 2018).

A third alternative that overcomes some of the aforementioned drawbacks is to use tools and techniques that measure physiological data. Patterns recorded using these techniques can help researchers understand the correlations between a task and changes in brain activity, without interrupting participants' thought processes (Schooler and Engstler-Schooler 1990; Schooler, Ohlsson, and Brooks 1993). In addition, using physiological data removes variables that make performance-based evaluations imprecise. Recent studies have employed such methods as functional magnetic resonance imaging (fMRI) (Fincham et al. 2010; Siegmund et al. 2017, 2020), near-infrared spectroscopy (NIRS) (Ikutani and Uwano 2014), eye-tracking (Bednarik and Tukiainen 2006; Lin et al. 2016), magnetoencephalography (MEG) (Jensen and Tesche 2002), or electroencephalogram tools (EEG) (Fritz et al. 2014; Crk, Kluthe, and Stefik 2016; Kosti et al. 2018). Researchers are particularly embracing on-the-scalp EEG devices, which when paired with software packages to assist signal processing, analyses and visualisation, give researchers an easy to use, non-invasive and affordable way to record and measure human brain activities.

In this paper, we seek to expand our prior work (Gopstein et al. 2017) by utilising EEG technology to determine if the presence of an atom in code influences the amount of cognitive workload experienced by study participants. Precisely, we test the premise that atoms will impose a higher cognitive load on participants, which will be reflected in the patterns of neuron oscillation detectable in EEG readings. We took the six atoms that had the highest effect sizes in our prior study (Gopstein et al. 2017), and used a non-invasive EEG headset to record the brain activities of 14 participants. We then analysed the participants' EEG data to see how strongly the changes in participants' cognitive load correlate with a particular atom type.[1] The following research questions guided our assessment of the data.

(RQ1) Do measurements of cognitive load differ based on the question type (obfuscated vs. clarified) being studied by the participant?

(RQ2) Do all atoms affect levels of cognitive load in the same way?

(RQ3) Do the levels of cognitive load correlate with performance-based measures, such as accuracy, reaction time and self-reported levels of confidence and difficulty?

We found that: (1) there was a difference in the degree of cognitive load induced by atom type; some atom types varied significantly, while others did not differ at all, (2) there was a discrepancy between participants' performance measures and their self-reported answers about a question's level of difficulty or their confidence in their responses, and (3) there was a clear correlation between participant reaction times and the cognitive load observed in the alpha band in the frontal region, which suggests that the alpha band in the frontal region could be useful for monitoring cognitive demand during program comprehension tasks.

In essence, our results suggest that some atoms did significantly increase cognitive load in our participants, while others had no effect at all. We speculate this variation can be attributed to two possible options: either the source of confusion in an atom is not caused by higher cognitive load, but may instead be caused by other factors, such as misreading the question, or the clarified snippets are as cognitively demanding as the obfuscated ones. One explanation of the latter might be that the transformation process introduced side effects and comprehension difficulties. These results affirm that the transformation process needs to be done thoughtfully and may require an understanding of how the atom relates to the syntax of the rest of the snippet. In a broader sense, our study shows that program comprehension studies must consider more human factors, such as a tendency to forget certain programming rules or overlook/underestimate what the code is asking them to do.

2. Background

In this section, we provide background information about our prior work (Section 2.1), as well as an overview of EEG measurement techniques (Section 2.2).

2.1. Atoms of confusion

We demonstrated in Gopstein et al. (2017) that confusion in software programs could be reduced by identifying and removing the smallest possible pieces of code that could influence developer perceptions. An example of an atom transformation is shown in Figure 1. The snippet (a), labelled as obfuscated, is functionally equivalent to snippet (b), labelled as clarified, but the atom has been removed.

We identified atom candidates from the winning programs of the International Obfuscated C Code Contest (IOCCC).[2] We eliminated the non-deterministic features in these programs that were impossible for humans to reliably predict, such as the `rand()` function, and features that were not portable across different computer architectures. The resulting 19 atom candidates are shown in Table 1 in Gopstein et al. (2017).

Next, we tested our hypothesis that *the existence of an atom in a code snippet would cause more errors in participants' responses than the snippet's transformed counterpart* by conducting a user study. A total of 73 participants reviewed a series of both obfuscated and clarified snippets, presented in a randomised order, and were asked to report the anticipated output for each one. Responses were graded for both accuracy and speed, and results confirmed that 15 of the candidates could be classified as atoms.[3] As an example, only 52% of all participants could correctly report the output of Figure 1(a). However, after removing this code pattern, participants' accuracy increased significantly for the transformed snippet (Figure 1b).

While our earlier study proved the existence of atoms, in this paper we seek insight on *why* these code patterns create confusion. The next section explains the measurement strategy chosen for the current study – collection and interpretation of brainwaves using an EEG device.

2.2. Electroencephalogram measurement

An EEG device measures the electrical field created by the activation of neurons, a specialised type of brain cell designed to transmit information to other nerve cells, or to muscle or gland cells. As information is transmitted to other neurons, tiny electrical fields are created. When a large number of neurons are activated in synchrony, the magnitude from the summed waveforms – in the order of μV – can be recorded by placing an EEG device on the scalp.

Brainwave signals are collected through an EEG device's channels (sensors) that are placed at locations following the international 10–20 system. In our study, we use four channels (F3, F4, F7 and F8) in the frontal region and two channels (P7 and P8) in the parietal region, which are the available channels in those regions from the EEG headset we use (shown in Figure 2).

Many studies have shown evidence that these inexpensive EEG headsets can reliably produce outcomes that are consistent with those predicted by existing theories. For example, Galán and Beal (2012) used a nine-channel EEG headset to measure engagement and workload readings for 16 college students as they solved SAT math problems. They found that the readings could distinguish between easy and hard problems with an accuracy rate of 87% and 83%, respectively. Rostami et al. (2015) used an EEG headset with only one channel and was able to find that mental effort – a reading reported by the EEG headset – was consistent with the observation notes of the facilitators during the experiment.

Table 1 lists studies related to cognitive tasks, along with their related frequency bands, regions or channels. We listed the range of frequency bands in each study when available because frequency ranges do vary, as do analysis techniques (which are not listed in the table). Nevertheless, what emerges from these studies is the importance of alpha and theta bands in brain imaging, as well as the relevance of brainwaves in the frontal and parietal regions to cognitive-related studies like ours.

```
void main() {
    char V1 = 2["qwert"];

    printf("%c\n", V1);
}
```

(a) Obfuscated

```
void main() {
    char V1 = "qwert"[2];

    printf("%c\n", V1);
}
```

(b) Clarified

Figure 1. Example of *Reversed Subscripts* snippet pair: (a) Obfuscated and (b) Clarified.

Table 1. Cognitive activities with corresponding frequency bands[†] and regions (C = central, P = parietal, T = temporal, F = frontal).

Cognitive activity	Frequency band (range)	Region (channel)	Reference
Program comprehension	θ (4–8 Hz)	C (uncertain)	Kosti et al. (2018)
	α and θ (varied by subject)	AF3, F7, F3, FC5	Crk, Kluthe, and Stefik (2016)
Arithmetic	β (16–24 Hz)	P (P3, P4)	Ray and Cole (1985)
	θ (3.5–7.5 Hz)	F (F3, Fz,[‡] F4) and P(P3, Pz,[‡] P4)	Sammer et al. (2007)
Attention	α (8–15 Hz)	P (P3, P4)	Ray and Cole (1985)
	α (8–13.5 Hz)	P (P7, P8), PO (PO3, PO4), and O (O1, O2)	Kelly et al. (2006)
Working Memory	θ (7–8.5 Hz)	F (MEG study, no channel)	Jensen and Tesche (2002)
	α (9–12 Hz)	Pz,[‡] O2, CP5, T8	Jensen et al. (2002)
	α (6–10 Hz) and θ (4–6 Hz)	Anterior (F3, F4, F7, F8, Fz[‡])	Krause et al. (1996)
	α (varied by subject)	F (F3, F4, FC5, FC6), C(C3, C4, FC1, FC2, CP1, CP2), P (P3, P4, CP5, CP6), and T (T3, T4)	Klimesch et al. (1999)
	θ (3.5–7.5 Hz)	F (F3, F4, FC5, FC6), C (C3, C4, FC1, FC2, CP1, CP2)	Klimesch et al. (1996)

[†] The exact frequency ranges are shown because they are defined differently. [‡] Z is the centre line, between odd and even numbers shown in Figure 2.

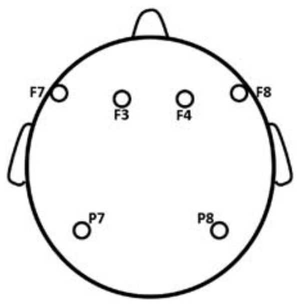


Figure 2. Location of six channels used in this study.

3. Related work

In this work, we draw from previous research using two methods of measuring comprehension – one based on the performance of study participants and the other based on physiological data drawn from the measurement of participants' brain activities. The following studies were influential in shaping our research questions, experimental design and interpretation of results.

3.1. Performance-based methods for measuring comprehension

The most common way to measure program comprehension is to let a programmer perform a task, such as estimating code output, finding or fixing bugs, adding additional features, and so on. Results are based on how well the task is completed, and the programmers' perception of its difficulty. In our study, we call these types of measurements performance measures.

Several studies (Shneiderman 1977; Soloway and Ehrlich 1984; Weidenbeck 1986; Pennington 1987b) equate comprehension with the amount of source code a programmer can recall. Unfortunately, the limited capacity of an individual's working memory (Miller 1956) makes testing by recall unreliable and unscalable. For example, when dealing with large, complex programs, recall measures may be low for both novices and experts (Miller 1956; Adelson and Soloway 1985). Another performance measure, called 'fill-in-the-blank,' asks participants to study a piece of code and then complete the portions that have been removed (Green 1977; Soloway and Ehrlich 1984). Wiedenbeck and Ramalingam (1999) used a hybrid of both methods in experiments where participants studied a program and then answered a series of questions without referring back to the source. Though not based on rote memorisation, the method still depends on the amount of information a programmer can retain.

These performance measures might be enough for laboratory experiments, but understanding software in the real world requires more than just information retention (Letovsky 1987). An alternative approach would be to ask developers to predict the outcome of a piece of code, to add new functions, or fix a problem in an existing program. Brooks (1977) used a set of talk-aloud protocols from a single programmer to construct a cognitive computer agent capable of predicting what results programmers might achieve. However, this type of simulation can be time consuming, subjective, and limited in scope. Though some researchers consider it more reliable (Dunsmore and Roper 2000), it does not negate the need for a comprehension measurement that is less reliant on the skills of the participant.

3.2. Physiological-based methods for measuring comprehension

As performance-based methods require a conscious response from all participants, the test itself could create cognitive demands capable of interfering with completion of the task. Therefore, recording a participant's physiological reactions (such as heartbeat, skin conductivity, pupil size or blood oxygen level) when engaging in cognitive processes can perhaps give a truer measurement. One common physiological method gathers data of brain images as participants perform cognitive tasks, such as program comprehension, and notes the changes from a resting state.

Using such brain imaging techniques, researchers have found cognitive functions can have local effects in specific regions of the brain. Previous studies have found that the frontal region is associated with cognitive activities, such as mental calculation (Dehaene and Cohen 1995), nonautomatic (i.e. nontrivial) calculation (Burbaud et al. 1999), memory recall (Gruber, Keil, and Müller 2001), word search (Yamamoto and Matsuoka 1990), and encoding and retrieval of long-term memory. Other studies associate the parietal region with number processing ability (Dehaene et al. 2003), arithmetic performance and processing (Barnea-Goraly et al. 2005; Andin et al. 2015) and development of mental arithmetic (Rivera et al. 2005). Based on this large body of work, we were able to interpret our brainwave readings of participants reviewing code snippets in terms of whether the presence of atoms may be interfering with the associated cognitive processes in different brain regions (Section 5.2).

Another way to identify the influence of cognitive activities on brainwaves is to analyse changes of power in each frequency band from the brain signals. Several studies (Sergeant, Geuze, and Van Winsum 1987; Klimesch et al. 1990; Krause et al. 2000; Kelly et al. 2006) have associated the alpha band with attention and found that alpha band power tends to decrease when participants perform cognitively demanding tasks. Also, studies of memory recall performance found a positive correlation to the power of upper alpha (10–12 Hz) (Vogt, Klimesch, and Doppelmayr 1998), and indicate cognitive activities can create fluctuations in the theta frequency band. Though Yamamoto and Matsuoka (1990) reported that long lasting theta waves recorded by EEG devices tended to correlate with participant reports of stress, Sammer et al. (2007) and Klimesch (1999) showed that theta band power was also positively related to mental workload.

Recently, a few studies have applied EEG measurements specifically to program comprehension tasks. In Kosti et al. (2018), the authors found that comprehension tasks were more cognitively demanding than syntax tasks, and that the theta frequency band was the most distinguishable biomarker between the two types of tasks. Crk, Kluthe, and Stefik (2016) found that change of magnitude in the theta frequency band was also the best indicator of programming expertise when testing five different student groups. Fritz et al. (2014) used multiple physiological sources, including eye-tracking and EEG data, to train a Naive Bayes classifier to analyse task difficulty. Their work offers a model for combining physiological methods to better assess levels of comprehension.

4. Experiment design

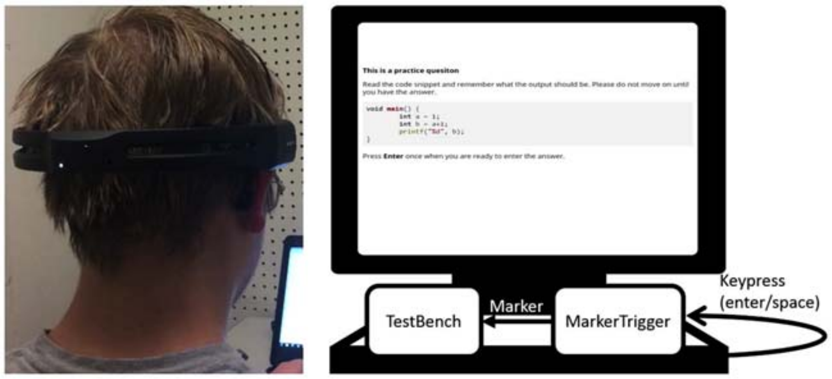
In this section, we describe our IRB-approved experiment. We explain our instrument setup (Section 4.1), describe the participants recruited (Section 4.2), and detail the procedure utilised to record and evaluate data (Section 4.3).

4.1. Instrument setup

To avoid fatigue, we chose to test only the six most confusing atoms – Change of Literal Encoding, Preprocessor in Statement, Logic as Control Flow, Post Increment/Decrement, Type Conversion, and Assignment as Value – from the list of atom candidates (Gopstein et al. 2017). For each atom, we selected four questions – two obfuscated and two clarified – for a total of 24 code snippets.[4]

The study instrument used includes two components. The first is a web application to present the code snippets to the participants and collect data such as snippet types (obfuscated or clarified), reaction times, answers to each question, and self-reported levels of difficulty and confidence in each response. The second is an EEG headset used to record brainwaves. The headset can record and transmit 10 channels (Figure 2) of raw EEG signals through a Bluetooth connection to a recording software on our computer called TestBench. We chose the highest possible sampling rate of 256 Hz for the device. Since the two components collected data separately, we had to add markers in the raw EEG data to indicate when a participant would be reading a particular code snippet and when the snippet would disappear from the screen. We developed a program called MarkerTrigger to capture two types of keystrokes, those made when using the space bar (indicating the beginning of the experiment) and the enter key (indicating the end of an event, such as finishing a question, or entering an answer).

Before the experiments, we installed all software applications and a Bluetooth receiver on a Dell laptop with a 15-inch screen running Windows 7. During the experiment, we opened the web application in Chrome and displayed one code snippet at a time in full screen mode. We reduced any artefacts that could be caused by muscle movements by removing the mouse and disabling the touchpad so all input had to be entered



This is a practice quesiton

Read the code snippet and remember what the output should be. Please do not move on until you have the answer.

```
void main() {
        int a = 1;
        int b = a+1;
        printf("%d", b);
}
```

Press **Enter** once when you are ready to enter the answer.

Figure 3. The experiment setup. All software applications were running in the background. Participants only saw the browser window.

through the keyboard. Figure 3 shows a participant during the experiment (left) and the computer setup (right).

4.2. Participants

We recruited 14 students, 3 female and 11 male, from Pennsylvania State University and Bucknell University. Two participants were graduate students and 12 were undergraduate students at the time of data collection. The participants' majors included Computer Science (5), Information Sciences and Technology (4), Computer Engineering (2), Mechanical Engineering (2), and Education (1). In compliance with our enrollment criteria, all participants had taken at least one semester of a C/C++ programming course. In the pre-screening survey, all indicated that they were healthy and not taking any medication.

4.3. Procedure

The order of events in the experiment, as shown in Figure 4, was largely controlled by the participants, allowing them to work at their own pace. Each participant first met individually with the researcher in an isolated room to learn about the purpose of the study. After informed consent was obtained, the participant was instructed to adjust the height of the chair, the screen angle of the laptop computer, and the distance between the participant and the computer. This is shown as the Introduction phase in Figure 4. Next, the Bluetooth EEG headset was placed on the participant's head. The reasearcher made adjustments to ensure that all 10 channels of interest were showing strong signals. The participant then read the on-screen instructions and was given one practice question to get familiar with the interface of the web application, including how to control the sliders used for rating difficulty and confidence levels by pressing the `f` and `j` keys, for left and right, respectively. This is the Interface Practice phase in Figure 4.

When the participants were ready, they were presented 24 code snippet questions, one at a time, in a random order. Before each code snippet, the participant was shown a screen with the message 'Relax' that stayed on for 10 s, cueing a brief pause (the Relax phase). The EEG data from the relax period was used as a baseline for comparison with the data from the program comprehension phase. Our analyses in Section 5.2 are based on the differences between these two time periods. After reading each code snippet, the participant took

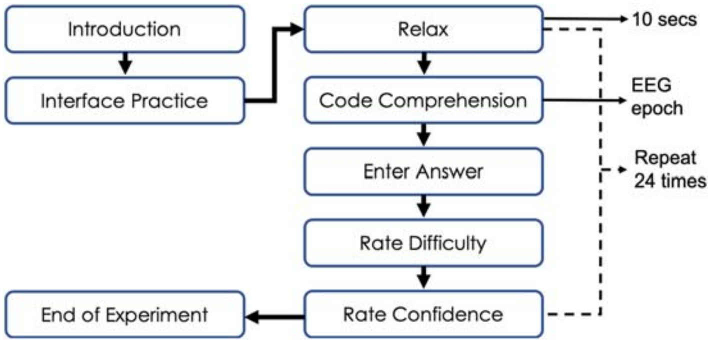


Figure 4. List of experiment events in sequence. Relax was a timed event (10 s). All others were controlled by the participants.

some time to interpret the code (the Code Comprehension phase), then hit the enter key to proceed to the next screen, where the participant typed the answer using the keyboard (the Enter Answer phase). The Code Comprehension and Enter Answer phases were separated in order to eliminate any noises from muscle movement when a participant entered an answer.

Two separate screens with range sliders followed each question, allowing participants to rate the difficulty of the preceding question and their level of confidence in their answers (the Rate Difficulty and Rate Confidence phases). The design of the slider was inspired by the work in Freyd (1923), where the authors stated that a graphic rating method can provide reasonable reliability while freeing the rater from 'direct quantitative terms.' In our experiment, participants moved the slider with the `f` and `j` keys so both hands could remain on the keyboard with no arm movements. The range of the slider was between 1 and 10 and the descriptive terms were 'easy' and 'difficult' on either side of the difficulty slider, and 'unsure' and 'very confident' on the confidence slider. At the end, each participant also filled out a demographic survey that asked their age, whether they were taking any medication, and whether they had any disorders, like insomnia, that could affect cognition.

4.4. EEG signal processing

We used an open-source application (Tadel et al. 2011) for brain imaging related analyses and visualisation, and Matlab to process the raw EEG data. The EEG signal data were obtained from the recording application (provided by the manufacturer) that converted and scaled measured analog signal to digital signal. No additional converting or scaling was done by the research team before signal processing. Following the headset manufacturer's recommendation, we chose to use a 0.16-Hz first-order high pass filter to remove the background signal. Because the frequency bands (alpha and theta) that we used in this study are both under 40 Hz, we first applied a band-pass filter between 0.16 and 40 Hz. We then used the filter as described in Melia et al. (2014) to remove peaks and spikes, which typically represent noise. The remaining amplitudes were limited to between ±200 μV. These procedures were conducted to ensure that any changes in brainwaves could be associated with cognitive load.

5. Results and analysis

In this section, we analyse two types of data gathered in the study: performance-based data (Section 5.1), which includes measures of participants' accuracy and speed of response, and physiological-based data (Section 5.2), which are the measurements recorded by the EEG device. In our analysis of both data sets, gathered using the procedure described in Section 4, we base our findings on responses from 10 of 14 participants.[5] The results were obtained from averaging participants' response and EEG signal data.

5.1. Performance-based data analyses

In this section, performance measures and perceptions of obfuscated and clarified code are analysed. We hypothesise that, when solving obfuscated questions, participants are more likely to get them wrong, spend more time working on them, perceive them to be more difficult, and feel less confident about their answers.

5.1.1. Accuracy and reaction time

To analyse performance, we first graded participants' answers and gave a score of 1 for a correct response or 0 for an incorrect response. In cases where the code snippet should output two variables, both values needed to be correct to receive a score of 1. Therefore, each participant received a score between 0 (all wrong) and 24 (all correct). From the scores, we then calculated participants' accuracy rates.

We separated all questions by atom type in order to identify any significant differences. Table 2 shows the breakdown of participant accuracy rates and reaction times for each of the atoms tested. A paired *t*-test was used to determine whether the differences between obfuscated and clarified questions were significant for each atom type.

Table 2. Comparison of accuracy rate and reaction time by atom type. Asterisk (*) indicates a significant difference between obfuscated (O) and clarified (C) questions.

Atom type	Accuracy rate				Reaction time in ms			
	O	C	t(df)	*p*	O	C	t(df)	*p*
Logic as Control Flow	10%	60%	−4.7434(9)	0.0011*	39561.4	30319.4	1.2223(9)	0.2526
Type Conversion	25%	85%	−4.8107(9)	0.001*	18652.55	37643.65	−6.7386(9)	0.0001*
Preprocessor in Statement	10%	90%	−9.798(9)	4.24E−06*	31563.2	16275.35	3.6186(9)	0.0056*
Assignment as Value	15%	75%	−4.8107(9)	0.001*	15082.95	21372.15	−1.4184(9)	0.1898
Post-Increment/Decrement	30%	80%	−2.7386(9)	0.0229*	23206.5	19788.3	0.578(9)	0.5775
Change of Literal Encoding	0%	90%	−13.5(9)	2.81E−07*	16443.2	11343.6	1.2635(9)	0.2382

As shown in Table 2, participants answered clarified questions for all atom types with significantly higher accuracy than their obfuscated counterparts (all p<.05). For reaction time, participants spent more time solving four types of obfuscated questions (Logic as Control Flow, Preprocessor in Statement, Post-Increment/Decrement and Change of Literal Encoding), though only the reaction time for Preprocessor in Statement was significantly different. By contrast, clarified questions in Type Conversion and Assignment as Value took participants more time to complete than their obfuscated counterparts. For Type Conversion, the time for participants solving clarified questions was significantly higher than the time spent on the obfuscated version.

Key Takeaway: Consistent with our previous work (Gopstein et al. 2017), our participants' accuracy in solving clarified questions was significantly better than when solving obfuscated questions. However, when reaction time was measured by atom type, only two registered significant differences. Obfuscated questions with the Preprocessor in Statement atom took significantly *more* time, while those with the Type Conversion atom took significantly *less* time.

5.1.2. Perceived confidence and difficulty levels

We next compared participants' perceptions as to the difficulty of the questions and their confidence in their answers. Table 3 shows the results of paired t-tests by each atom type. For *perceived confidence level*, participants were more confident in answering clarified questions in all atom types except for Assignment as Value. Among the remaining five atoms, four showed significant differences (indicated by an asterisk after the p value in Table 3), while Type Conversion showed almost no difference. For *perceived difficulty level*, participants reported higher levels of difficulty for obfuscated questions in all atoms except Assignment as Value and Type Conversion. However, only Post-Increment/Decrement and Change of Literal Encoding exhibited significant differences.

Based on the data above, the results of two atoms, Assignment as Value and Type Conversion, seem to consistently run counter to the data of other atoms. Clarified questions with these atoms took longer to solve, and participants were less confident about their responses, but the snippets were not perceived as difficult. We address why these atoms might be outliers in Section 5.2, with the help of physiological data.

Correlation among Performance Measures: To see the relationships among these variables, we calculated the correlation between accuracy, reaction time, and perceived confidence and difficulty. The results are illustrated in Table 4. We can draw the following conclusions: (a) when participants gave a higher rating in their confidence, the reaction time tended to be shorter ($r = -0.3426,\ p < .05$), and the accuracy tended to be higher ($r = 0.2235,\ p < .05$), and (b) when a question was deemed more difficult, the participants spent more time on it ($r = 0.3137,\ p < .05$), felt less confident in their answers ($r = -.7403,\ p < .05$), and were less likely to answer it correctly ($r = -0.2338,\ p < .05$).

Table 5 provides a summary of how atoms affect accuracy, reaction time, perceived confidence and difficulty levels. The Exp column indicates whether the result was expected, e.g. the accuracy of clarified questions was higher than obfuscated questions. The Sig column indicates for each measure whether the difference between obfuscated and clarified questions was significantly different. The ideal result for each measure is [✓, ✓] in both Exp and Sig. However, except for accuracy, none of the measures has [✓, ✓] across all atom types. Furthermore, the results from the accuracy measure were insufficient to affirm that participants comprehended clarified questions better than obfuscated ones. For example, for Type Conversion, even though participants answered clarified questions with higher accuracy, they did not perceive the obfuscated questions to be more difficult, and they spent significantly *less* time in solving these questions (shown as [✗, ✓]). Therefore, the performance measures alone are not sufficient to draw informed conclusions.

Key Takeaway: Among the four performance measures studied, three – reaction time, perceived confidence, and perceived difficulty – were not effective at indicating differences between clarified or obfuscated code snippets. One possible reason is that the

Table 3. Comparison of perceived confidence and difficulty by atom type. Asterisk (*) indicates a statistically significant difference. O stands for obfuscated and C stands for clarified.

Atom type	Confidence				Difficulty			
	O	C	t(df)	p	O	C	t(df)	p
Logic as Control Flow	5.15	6.75	−2.6264(9)	0.0275*	4.25	3.2	1.4198(9)	0.1894
Type Conversion	6.3	6.7	−0.937(9)	0.3732	2.65	2.75	−0.175(9)	0.865
Preprocessor in Statement	5.75	7.15	−2.3515(9)	0.0432*	3.5	2.75	1.6948(9)	0.1244
Assignment as Value	7.7	5.75	1.5735(9)	0.1501	2.25	2.85	−0.8742(9)	0.4048
Post−Increment/Decrement	7.35	8.3	−2.3486(9)	0.0434*	2.4	1.95	2.5861(9)	0.0294*
Change of Literal Encoding	6.45	7.35	−2.7848(9)	0.0212*	2.9	2.35	3.1608(9)	0.0115*

Table 4. Correlation between accuracy, reaction time, confidence, and difficulty levels. Asterisk (*) indicates a statistically significant correlation.

	Accuracy	Reaction Time	Confidence	Difficulty
Accuracy	1			
Reaction Time	−0.0559	1		
Confidence	0.2235*	−0.3426*	1	
Difficulty	−0.2338*	0.3137*	−0.7403*	1

Table 5. Summary of the effect of atoms on accuracy, reaction time, perceived confidence and difficulty levels. Exp means the result is expected; Sig means the difference between the results is significant. Unexpected but significant results are in red and circled, like ✗ and ✓.

Atom type	Accuracy		Reaction time		Confidence		Difficulty	
	Exp	Sig	Exp	Sig	Exp	Sig	Exp	Sig
Logic as Control Flow	✓	✓	✓	✗	✓	✓	✓	✗
Type Conversion	✓	✓	✗	✓	✓	✗	✗	✗
Preprocessor in Statement	✓	✓	✓	✓	✓	✓	✓	✗
Assignment as Value	✓	✓	✗	✗	✗	✗	✗	✗
Post Increment/ Decrement	✓	✓	✓	✗	✓	✓	✓	✓
Change of Literal Encoding	✓	✓	✓	✗	✓	✓	✓	✓

participants lacked sufficient programming experience to make informed judgments about the complexity of the snippets they reviewed. While accuracy rates did vary as expected, the findings cannot confirm if a correct answer was the result of a good guess, or if an incorrect response was simply a typographical error. Either way this judgement is based solely on participants' answers, and therefore not objective. By analysing the physiological data gathered from participants, we seek a less subjective dataset that can link the influence of atoms to measurable differences in brainwaves.

5.2. Physiological-based data analyses

In this section, we analyse EEG data obtained using the instrument and apparatus described in Section 4 to understand *whether* and *how* atoms can affect brain activities. Specifically, we wanted to see if brainwave patterns differed when participants solved obfuscated snippets. We first introduce the metrics we use and then answer the three research questions from Section 1.

5.2.1. Measuring cognitive load

We are interested in testing whether obfuscated questions induce higher cognitive load, as this can help us better understand why atoms affect comprehension. Cognitive load was measured based on the characteristics of neuron oscillation. Neurons oscillate constantly, whether a human is at rest or is performing a cognitive task. When neurons oscillate in synchrony (i.e. same or similar phase), the amplitude of the waveform is greater. The degree to which the oscillation of neurons synchronise can be affected by cognitive demand (Klimesch 1999; Antonenko and Niederhauser 2010), where higher cognitive demand is correlated with lower magnitude (desynchronisation). To measure event-related synchronisation, e.g. desynchronisation caused by an event such as code comprehension, a short period of time immediately preceding the task is normally used as a reference point. During this time, participants are typically asked to relax. By taking the average band power from the reference point and comparing it to when the participant performs a cognitive task, the amount of change can be calculated and analysed.

Event-Related Synchronisation/ Desynchronisation, or ERS/ERD, presented in Equation (1) (Pfurtscheller and Da Silva 1999) is commonly used to model the degree of changes in the magnitude of brainwave signals (average band power, or `Avg.BP`, in the equation), from a reference period (indicated by subscript `reference`) to a task period (indicated by subscript `task`).

$$\text{ERS (ERD)\%} = \frac{\text{Avg.BP}_{\text{task}} - \text{Avg.BP}_{\text{reference}}}{\text{Avg.BP}_{\text{reference}}} \times 100\%. \quad (1)$$

The reference period in our study is the Relax period preceding each code snippet while the task period is the Code Comprehension period that follows (see Figure 4). The average band power of each period is acquired by averaging the magnitude of a frequency band from these two periods separately. When the result of Equation (1) is positive, or the average band power during the task period is higher, the result indicates ERS, i.e. neuron oscillation is more synchronised during the Code Comprehension period than the rest period. By contrast, a negative result from Equation (1) indicates ERD, or neuron oscillation is more synchronised during the rest period than the Code Comprehension period. ERD has previously been shown to be correlated to high cognitive load (Klimesch 1999; Antonenko and Niederhauser 2010), so we used it in this study to mark which atoms created higher levels of cognitive demand.

In the following, we use our physiological-based data to answer the three research questions (RQ's) raised in Section 1. For each RQ, we first present our null (H_0) hypotheses, then we describe our experiment, results and takeaway.

5.2.2. RQ1: can EEG readings be used to distinguish between obfuscated and clarified questions?

To find an answer to this question, we use the experiment design summarised in Figure 5.

In this experiment, we treat all obfuscated code as the control group, and all clarified code as the treatment group. We first calculated the ERS/ERD values for each code snippet, with Equation (1) applied to alpha and theta frequency bands in the EEG signals. We refer to this as the *overall* result. We then performed a paired *t*-test to verify whether the ERS/ERD values were significantly different when participants were solving obfuscated questions as compared to clarified questions. In addition, we calculated results from Equation (1) by grouping the channels into frontal and parietal regions (Figure 2 in Section 2.2), along with a paired *t*-test for both frequency bands within each region. We refer to this as the results from the *frontal* and *parietal* regions.

Our findings are presented in Table 6. First, all values in $ERD_{obfuscated}$ and $ERD_{clarified}$ columns are negative. This indicates ERD instead of ERS, i.e. desynchronised neuron oscillations were detected during the task period. Since we expected a higher cognitive load during Code Comprehension, this is consistent with previous findings (Antonenko and Niederhauser 2010) where a higher cognitive load was associated with desynchronisation. Second, every *p* value, including those from overall and frontal/parietal regions, indicates that the differences between $ERD_{obfuscated}$ and $ERD_{clarified}$ are *not* significant. We further compared the relative changes in ERD values, or ERD_{diff}, as follows:

$$ERD_{diff} = ERD_{clarified} - ERD_{obfuscated} \qquad (2)$$

We expected that all ERD_{diff} would be greater than zero; if all obfuscated snippets cause higher cognitive load, then $ERD_{obfuscated}$ should be smaller than $ERD_{clarified}$. The values of ERD_{diff} are also shown in Table 6. Results show that in the alpha frequency band, all ERD_{diff} values are positive, but in the theta frequency band two values are negative. Prior work by Klimesch et al. (1997) showed that ERD in the alpha frequency band is often associated with processing semantic information. Therefore, ERD_{diff} in the alpha frequency band may be a better indicator of higher cognitive load during program comprehension.

Key Takeaway: Data from both frequency bands suggests a higher cognitive load during program comprehension. There is, however, no overall evidence that every obfuscated snippet induces a larger cognitive load than every clarified snippet. This is particularly troublesome because several prior studies have shown that stronger brainwave signals in the theta frequency are associated with higher cognitive load (Krause et al. 2000; Crk, Kluthe, and Stefik 2016; Kosti et al. 2018). Our results so far suggest that either none of the obfuscated snippets correlate with higher cognitive load or only some do. This propelled us to take a look at the data by individual atom types.

5.2.3. RQ2: do all atom types affect EEG readings in a similar way?

A summary of our experiment design is shown in Figure 6. To answer this question, we first separated the ERD values by atom type and compared ERD_{diff} in both frequency bands (Section 5.2.3.1). Then we grouped ERD_{diff} in all channels into frontal and parietal regions (Section 5.2.3.2), and, finally, we looked at atoms that had no effect on ERD_{diff}.

5.2.3.1 Overall significance:. We first separated the ERD values by atom type, and compared ERD_{diff} in alpha and theta frequency bands. The results are shown in Table 7. As explained in RQ1, ERD_{diff} is expected to be greater than zero, which is shown by a ✓ in the Expected columns in the table. If a paired *t*-test shows a significant difference between $ERD_{obfuscated}$ and $ERD_{clarified}$, this is indicated by a ✓ in the Significant columns.[6] Insignificant differences are not shown.

Control: Code snippet containing a single atom, i.e., obfuscated snippet.
Treatment: A version of the control code transformed to remove the atom, i.e., clarified snippet.
Null Hypothesis H_0: Question types (obfuscated vs. clarified) have the same effect on brain activities.

Figure 5. Summary of experiment design, RQ1.

Table 6. Comparison of ERD by question types in overall, frontal region and parietal region.

Frequency band	Region	*ERD_{obfuscated}*	*ERD_{clarified}*	*ERD_{diff}*	*p*
Alpha	Overall	−32.19629	−31.1932	1.00309	0.8852
	Frontal	−34.2	−32.74	1.46	0.8387
	Parietal	−35.52475	−33.06558	2.45917	0.7346
Theta	Overall	−29.6218	−31.90715	−2.28535	0.485
	Frontal	−32.443	−34.973	−2.529	0.513
	Parietal	−32.886	−32.557	0.329	0.943

Control: Code snippet containing a single atom, i.e., obfuscated snippet.
Treatment: A version of the control code transformed to remove the atom, i.e., clarified snippet.
Null Hypothesis H_0: Different atom types have the same effect on brain activities, such that there are no significant changes in ERS/ERD values of any atom type.

Figure 6. Summary of experiment design, RQ2.

Table 7. ERD_{diff} by atom type. The expected columns show whether $\text{ERD}_{\text{obfuscated}}$ is smaller than $\text{ERD}_{\text{clarified}}$, i.e. ERD_{diff} is positive and thus expected. The significant columns show whether the difference between $\text{ERD}_{\text{obfuscated}}$ and $\text{ERD}_{\text{clarified}}$ is significant. For ease of reading, insignificant differences are not shown. Unexpected but significant results are in red and circled, like ✗ and ✓.

Atom type	Alpha		Theta	
	Expected	Significant	Expected	Significant
Logic as Control Flow	✓	✓	✓	✓
Type Conversion	✗	✓	✗	✓
Preprocessor in Statement	✓	✓	✓	✓
Assignment as Value				
Post-Increment/ Decrement				
Change of Literal Encoding				

The results, as summarised in Table 7, show that three out of six atom types, Logic as Control Flow, Type Conversion and Preprocessor in Statement, have a significant ERD_{diff} in both frequency bands. Surprisingly, for Type Conversion, the clarified questions incurred a significantly higher cognitive load than the obfuscated ones in both frequency bands, shown as [✗, ✓] in the Expected and Significant columns. We next discuss this unexpected result.

Type Conversion. An example snippet pair for this atom is shown in Figure 7. The obfuscated snippet in Figure 7(a) requires converting the value of the integer variable `V1` to a type of `char`. The result is the same as `V1%256` in Figure 7(b), due to overflow. The `char` type in C only has 8 bits, therefore a maximum ASCII value of a character is 255. Since an `int` variable contains more bits than a `char`, any integer greater than 255 will be rounded up to `V1%256`, or `V1%2`^{8}.

Participants may overlook the fact that in Figure 7(a), an integer is assigned to a character variable, and thus they did not perceive the obfuscated question to be difficult (also seen in Table 3). In turn, they printed the integer value directly. However, Figure 7(b) requires computing a modular operation `V1%256`, which would incur a higher cognitive load and result in a negative ERD_{diff}. Previous studies have associated ERD in the alpha frequency band with task difficulty (Petsche, Pockberger, and Rappelsberger 1984; Sterman et al. 1994; Gevins et al. 1997) and attention (Sergeant, Geuze, and Van Winsum 1987; Kelly et al. 2006). As Type Conversion also exhibited unexpected significance in the alpha frequency band, we speculate that participants underestimated the complexity of the code. As a result, their level of attention was reduced and so they did not notice that the code required a conversion.

5.2.3.2 Significance by region:. Next, we grouped ERD_{diff} in all channels into frontal and parietal regions. The results are shown in Table 8. The Exp columns show whether ERD_{diff} is greater than zero, as expected. The Sig columns show if there is a significant difference between $\text{ERD}_{\text{obfuscated}}$ and $\text{ERD}_{\text{clarified}}$. For ease of reading, insignificant differences are not shown.[7] The results show that the changes are more pronounced in the alpha frequency band than in the theta frequency band when cognitive load is expected to be higher. Cognitive load measures from two atoms (Type Conversion and Assignment as Value) are significantly different, but not in the way that we expected. The differences in cognitive load in two atoms (Post-Increment/Decrement and Change in Literal Encoding) were not significant. We explain briefly the possible reasons below for three of these four atoms, as Type Conversion has already been discussed above.

Atom with an Unexpected Significant Effect: Assignment as Value. An example snippet pair is shown in Figure 8. In Figure 8(a), the condition `(V1 = 0)` in the `if` statement is an assignment, and evaluating any assignment operation always results in a logical `true`. In Figure 8(b), the condition `if (V1)` requires that

```
void main() {
    int V1 = 261;
    char V2 = V1;
    printf("%d\n", V2);
}
```

(a) Obfuscated

```
void main() {
    int V1 = 261;
    char V2 = V1 % 256;
    printf("%d\n", V2);
}
```

(b) Clarified

Figure 7. Examples of type conversion atom: (a) Obfuscated and (b) Clarified.

Table 8. ERD_{diff} by atom type in frontal and parietal regions. The Exp columns show whether $ERD_{obfuscated}$ is smaller than $ERD_{clarified}$, i.e. ERD_{diff} is positive and thus expected. The Sig columns show whether the difference between $ERD_{obfuscated}$ and $ERD_{clarified}$ is significant. For ease of reading, insignificant differences are not shown. Unexpected but significant results are in red and circled.

Atom type	Frontal				Parietal			
	Alpha		Theta		Alpha		Theta	
	Exp	Sig	Exp	Sig	Exp	Sig	Exp	Sig
Logic as Control Flow			✓	✓	✓	✓	✓	✓
Type Conversion				fx1				
Preprocessor in Statement	✓	✓	✓	✓				
Assignment as Value			fx2					
Post-Increment/ Decrement								
Change of Literal Encoding								

participants evaluate it as `if (V1 != 0)`. The fact that participants correctly answered only 15% of the snippets like Figure 8(a), but 75% of the snippets like Figure 8(b), indicate that most overlooked the condition in `if (V1 = 0)` and read it as `if (V1 == 0)`. This misinterpretation is also reflected by an average reaction time of 15.1 s for obfuscated snippets, but 21.4 s for clarified snippets (Table 2). Therefore, participants mistakingly perceived the obfuscated snippets to be easier. This resulted in a negative ERD_{diff}.

Atoms with No Significant Effect: In Tables 7 and 8, Post-Increment/Decrement and Change of Literal Encoding had no effect on EEG data. For these two atoms, we offer the following reasoning.

Post-Increment/Decrement. An example snippet pair is shown in Figure 9. A post-increment operator is used to increase the value of a variable after the statement has been evaluated. In Figure 9(a), variable `V1`'s initial value `2` is first used in the expression `int V2 = 3 + V1++;` and then increased to `3`. In Figure 9(b), `V1++` is in a single statement, so there is no ordering issue that can potentially create confusion. From the drastically different accuracy rates (30% for obfuscated snippets and 80% for clarified snippets), and the significant differences in participants' perceived confidence and difficulty levels (Table 3), it is clear that participants had a hard time solving the obfuscated snippets, and correctly perceived that level of difficulty. However, the reaction time does not show a significant difference in how much time was spent on the snippet. The data suggests that the participants were aware of the ordering issue but were unable to solve the obfuscated snippets as accurately. They may have realised the existence of an atom, but lacked the knowledge to solve it. As a result, their answers were likely educated guesses, which could explain the insignificant differences in cognitive load in Tables 7 and 8.

Change of Literal Encoding. An example snippet pair is shown in Figure 10. In Figure 10(a), the integer variable `V1` is assigned an octal value `013`. However, when `V1` gets printed in the following statement, it should be displayed as a decimal value, indicated by `%d` in `printf`. In Figure 10(b), the integer variable `V1` is both assigned and printed as decimal. The obfuscated snippet looks very similar to the clarified one at first glance. In Figure 10(a), participants might not have been aware of the encoding and formatting differences. If participants were not able to distinguish between the two, the type of code snippet would have no influence on cognitive load during code comprehension, as shown by the accuracy rates (90% for the clarified snippets and 0% for the obfuscated snippets), and the insignificant differences in Tables 7 and 8.

Key Takeaway: Looking at ERD_{diff} by atom types confirms the premise that atoms affect brain activities differently. Some atoms create significantly different cognitive load between its obfuscated and clarified versions. However, there are also atoms where the difference is not significant, and where EEG data cannot distinguish between obfuscated and clarified snippets. It seems the ERD changes in the frontal region is where the difference is most appreciable.

5.2.4. RQ3: Do EEG readings correlate with participants' reaction time, perception of code difficulty and confidence in responses?

To answer this research question, we use the experiment design summarised in Figure 11.

This research question looks for the relationship between performance measures and the physiological

```
void main() {
    int V1 = 0;

    if (V1 = 0) {
        printf("true\n");
    } else {
        printf("false\n");
    }
}
```

(a) Obfuscated

```
void main() {
    int V1 = 0;

    if (V1) {
        printf("true\n");
    } else {
        printf("false\n");
    }
}
```

(b) Clarified

Figure 8. Examples of assignment as value atom: (a) Obfuscated and (b) Clarified.

```
void main() {
    int V1 = 2;
    int V2 = 3 + V1++;
    printf("%d␣%d\n", V1, V2);
}
```

(a) Obfuscated

```
void main() {
    int V1 =  2, V2;
    V2 = V1 + 3;
    V1++;
    printf("%d␣%d\n", V1, V2);
}
```

(b) Clarified

Figure 9. Examples of post-increment/decrement atom: (a) Obfuscated and (b) Clarified.

```
void main() {
     int V1 = 013;
     printf("%d\n", V1);
 }
```

(a) Obfuscated

```
void main() {
     int V1 = 23;
     printf("%d\n", V1);
 }
```

(b) Clarified

Figure 10. Examples of change of literal encoding atom: (a) Obfuscated and (b) Clarified.

data. Because $\mathrm{ERD}_{\mathrm{diff}}$ between the two question groups was not significant (RQ1) and there was no significant difference in reaction time, perceived confidence and difficulty levels, one might assume that solving the obfuscated snippets did not increase the cognitive load of the participants. Yet, the poor accuracy rate for obfuscated questions and $\mathrm{ERD}_{\mathrm{diff}}$ in some atom types suggest that at least a few of these questions challenged the cognitive load of participants, even if they were unaware at the time that they were working harder.

As shown in Table 9, the correlation between the ERD values and performance measures was only significant with reaction time ($p < .05$). Specifically, reaction time had a significant negative correlation (between -0.7166 and -0.9378) with ERD values across both frequency bands in overall, frontal and parietal regions. This strong correlation suggests that the more time participants spend solving a question, the higher the cognitive load will be.

We further broke down the reaction time by question types, i.e.obfuscated and clarified, as shown in Table 10. For obfuscated questions, ERD values were significantly correlated with reaction time across all frequency bands in all regions; for clarified questions, however, there was no significant correlation in either frequency band in the parietal region. The table also shows that the desynchronisation in the frontal region was highly correlated with reaction time. The lack of significant correlation in the parietal region for clarified questions could be attributed to the fact that the relevant cognitive activities of simple logic and arithmetic may be localised in the frontal region (Dehaene and Cohen 1995; Burbaud et al. 1999). As these activities are not present in the parietal region, it remains unaffected.

Key Takeaway: As ERD values were significantly correlated with reaction time, our results partially reject our null hypothesis H_0. In evaluating these findings, however, it is important to remember that two of the performance values, perceived confidence and difficulty levels, are self-ranked. Given that the participants in our study were students whose experience with C was somewhat limited, these results might suggest an inability to correctly judge complexity, or that participants had their own criteria for rating confidence and difficulty levels that differed from how much time they spent on a question.

5.3. Summary of results

It seems natural to hypothesise that when people are confused or challenged by a problem, they work harder mentally to solve the problem. In doing so, we expect to see an increase in their cognitive load, which should manifest itself in other ways as well, such as the amount of time spent on the problem (Haga, Shinoda, and Kokubun 2002). If one looks only at our results from the combined obfuscated vs. clarified groups (RQ1), however, it does not support these hypotheses. Despite scoring significantly lower in obfuscated compared to clarified questions, there was no significant aggregate difference in reaction time, self-reported confidence or assessment of difficulty. Likewise, our physiological data did not support the premise that 'all obfuscated snippets are cognitively demanding' (RQ2).

Control: Code snippet containing a single atom, i.e., obfuscated snippet.
Treatment: A version of the control code transformed to remove the atom, i.e., clarified snippet.
Null Hypothesis H_0: Accuracy, reaction time, perceived difficulty and confidence do not correlate with the ERS/ERD values.

Figure 11. Summary of experiment design, RQ3.

Table 9. Correlation between ERD values and accuracy, reaction time, confidence, and difficulty. Asterisk (*) indicates a statistically significant difference (*p*<.05).

Frequency band	Region	Accuracy	Reaction time	Confidence	Difficulty
Alpha	Overall	−0.082	−0.7568*	0.3444	−0.2323
	Frontal	0.1824	−0.8891*	0.5328	−0.5224
	Parietal	−0.2386	−0.7238*	0.2557	−0.0788
Theta	Overall	−0.2024	−0.6727*	0.2619	−0.1954
	Frontal	−0.0123	−0.7068*	0.28	−0.3676
	Parietal	−0.3622	−0.6501*	0.249	−0.0539

Table 10. Correlation between ERD and reaction time with different question types. Asterisk (*) indicates a statistically significant difference (*p*<.05).

Frequency band	Region	Clarified	Obfuscated
Alpha	Overall	−0.7544*	−0.9140*
	Frontal	−0.8440*	−0.8533*
	Parietal	−0.6063	−0.7652*
Theta	Overall	−0.7157*	−0.8079*
	Frontal	−0.7363*	−0.6778*
	Parietal	−0.4927	−0.7237*

Instead, our results suggest individual atoms will affect cognitive load in programmers to very different degrees. Therefore, using performance measures, as we did in our previous studies, does not provide enough information to capture the reason why programmers make mistakes. While the physiological data we gathered suggests that some atoms might be inherently mentally-demanding, others may be problematic for other reasons. Given the experience levels of our participants, we believe they struggled because they do not have the knowledge or experience to be aware of the challenges inherent in these snippets, or know how to address them.

Taken together, our data suggests that understanding how programmers interpret and understand code can best be achieved by utilising both approaches. Though EEG-based physiological measurement appears to provide more useful insights than the performance measures, it would be impossible to understand the psychology of program comprehension only from EEG signals. If a participant cannot discern any difference between an obfuscated snippet and a clarified one, there would be no neural signatures for us to discern. Rather than relying on one strategy, researchers should develop tests that can use the benefits of both. The performance measurement tells us whether the participants can perform program comprehension correctly, and the physiological measures tell us whether the task is cognitively demanding.

6. Discussion

While our results showed high cognitive load correlated to poor performance in a few instances, we were a bit surprised to see this correlation did not hold true in every case. There were enough instances where performance scores were poor but cognitive load was low to suggest that other factors, such as a lack of experience, or flaws in the way participants learned about certain operations might be the cause. Below are ways both types of performance barriers can be mitigated, particularly for novice developers.

Avoiding/mitigating atoms that have a high cognitive workload: For certain atoms, a higher cognitive load seems to positively correlate with higher error rates. Therefore, designers of programming tools, like integrated development environments (IDEs) or version control systems, can utilise this information to help developers avoid these atoms whenever possible. Modern IDEs already incorporate real-time notifications and warnings, so atoms with a high potential for taxing cognitive load could trigger an alert. The programmer can then modify the code accordingly.

Leveraging programming experience: Our findings suggest that novice programmers may have some blindspots that they are not even aware of. Modern IDEs allow users to customise the appearance and layout of the tools they use. But when it comes to how source code is constructed, these tools are not designed for users of all experience levels. Giving students or novice programmers the freedom to build programs any way they want could be setting them up to fail. Therefore, we suggest that designers build programming tools that account for the skill levels of the programmer by providing adaptive interfaces that can be made more flexible as the user gains experience.

Teaching programming languages: Atoms could be an impediment to a student's ability to master any new programming language, therefore it would be wise to identify what might qualify as an atom in other programming languages, such as Java and Python. Here again, tests utilising physiological data could help determine not only if an atom is confusing but also if the cause might be high cognitive load. Once these types are identified, programming concepts that include atoms could be supported for novices through scaffolded instruction. Students would learn a simpler alternative first and use that as a building block to mastering the language's more cognitively demanding counterparts.

7. Conclusion

In this paper, we explored the feasibility of using brainwave data collected from an EEG device as an alternative way to understand the code comprehension processes of programmers. We asked our participants to predict outputs for small snippets of code that contained 'atoms of confusion,' or minimal patterns that in previous studies had proved confusing to developers. We found that: (a) measurements of cognitive load are generally not significantly greater when solving obfuscated snippets as opposed to clarified ones, but there are a few individual atoms where the differences are significant, (b) measurements of reaction time, perceived confidence and perceived difficulty are not consistently reliable indicators of whether a code snippet should be labelled as confusing, and (c) readings from the frontal region of the brain reflect program comprehension-induced cognitive load better than those from the parietal region.

7.1. Future work

Building on our study, we see several promising approaches for furthering our understanding of the cognitive processes utilised in programming comprehension through the use of physiological data. One such direction could be to use EEG measurements to broadly categorise atoms by the cognitive activities they require. Atoms that tend to induce cognitive load could be identified this way, and then either be removed (e.g. by providing some type of cues to make the coding issue overt) or have its effects mitigated (e.g. by suggesting equivalent but less cognitively demanding programming features). A recent study by Peitek et al. (2018), which utilised an fMRI device to measure what areas of the brain were activated as participants completed comprehension activities, suggests such a categorisation study could be effective.

Another approach would be to collect physiological data from a different modality, such as eye-tracking or functional magnetic resonance imaging (fMRI), to provide additional data points for analyses as suggested by Fritz et al. (2014). Analysing eye-tracking data, along with data from an EEG device, could help researchers narrow the source of confusion from a code snippet to a statement.

Lastly, adding new measurement techniques can help deepen our overall understanding about program comprehension, which is still very limited. While programming style guides provide practical advice for writing code, the lack of accepted program comprehension models means there is no way to assess and predict the readability of a program. As more research studies harness physiological measurement tools, e.g. Lin et al. (2016); Fritz et al. (2014); Peitek et al. (2018), the data that emerges can hopefully be used to build evidence-based models/metrics that can be used to assess the readability of program code.

Notes

1. In the study, we use the phrase 'question type' when referring to obfuscated vs. clarified snippets, i.e. code snippets that contain an atom vs. code with the atom removed through transformation. We use 'atom type' when referring to specific code patterns, such as the *Reversed Subscripts* pattern shown in Figure 1.
2. https://www.ioccc.org/
3. To qualify as atoms in our test, the snippets containing atoms had to be significantly more confusing than their transformed equivalent. Atom candidates were rejected if either (1) the atom candidate was not confusing or (2) the transformation failed to remove confusion.
4. All questions used in the study can be found on our project page: https://atomsofconfusion.com/2016-snippet-study/questions.
5. We chose to exclude data from four subjects for two different reasons. For the first two, we found the brainwave recordings contained excessive noise and could not be used. We corrected this problem by replacing the laptop used to collect data. Data from two other participants were excluded because, for unknown reasons, the recordings did not have all the expected markers.
6. The raw ERD data that is used to produce Table 7 is available in Table 1 on http://martinyeh.com/papers/eeg-appendix.html
7. The raw ERD data are provided in Tables 2 and 3 on http://martinyeh.com/papers/eeg-appendix.html.

Acknowledgements

The authors would like to thank Nicole Wang for her help with data collection.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported in part by a Committee on Research and Creative Works (CRCW) award at the University of Colorado, Colorado Springs, and by the National Science Foundation under Award No. 1444827.

ORCID

Martin K.-C. Yeh http://orcid.org/0000-0002-5630-1633

References

Adelson, B., and E. Soloway. 1985. “The Role of Domain Expence in Software Design.” *IEEE Transactions on Software Engineering* 11 (11): 1351–1360.

Andin, J., P. Fransson, J. Rönnberg, and M. Rudner. 2015. “Phonology and Arithmetic in the Language-Calculation Network.” *Brain and Language* 143: 97–105.

Antonenko, P. D., and D. S. Niederhauser. 2010. “The Influence of Leads on Cognitive Load and Learning in a Hypertext Environment.” *Computers in Human Behavior* 26 (2): 140–150.

Atman, C. J., and K. M. Bursic. 1998. “Verbal Protocol Analysis as a Method to Document Engineering Student Design Processes.” *Journal of Engineering Education* 87 (2): 121–132.

Barnea-Goraly, N., S. Eliez, V. Menon, R. Bammer, and A. L. Reiss. 2005. “Arithmetic Ability and Parietal Alterations: a Diffusion Tensor Imaging Study in Velocardiofacial Syndrome.” *Cognitive Brain Research* 25 (3): 735–740.

Bednarik, R., and M. Tukiainen. 2006. “An Eye-Tracking Methodology for Characterizing Program Comprehension Processes.” In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications*, 125–132.

Brooks, R. 1977. “Towards a Theory of the Cognitive Processes in Computer Programming.” *International Journal of Man-Machine Studies* 9 (6): 737–751.

Burbaud, P., O. Camus, D. Guehl, B. Bioulac, J. M. Caillé, and M. Allard. 1999. “A Functional Magnetic Resonance Imaging Study of Mental Subtraction in Human Subjects.” *Neuroscience Letters* 273 (3): 195–199.

Corritore, C. L., and S. Wiedenbeck. 1991. “What Do Novices Learn During Program Comprehension?.” *International Journal of Human-Computer Interaction* 3 (2): 199–222.

Crk, I., T. Kluthe, and A. Stefik. 2016. “Understanding Programming Expertise: An Empirical Study of Phasic Brain Wave Changes.” *ACM Transactions on Computer–Human Interaction (TOCHI)* 23 (1): 2.

Dehaene, S., and L. Cohen. 1995. “Towards An Anatomical and Functional Model of Number Processing.” *Mathematical Cognition* 1 (1): 83–120.

Dehaene, S., M. Piazza, P. Pinel, and L. Cohen. 2003. “Three Parietal Circuits for Number Processing.” *Cognitive Neuropsychology* 20 (3–6): 487–506.

Dunsmore, A., and M. Roper. 2000. “A Comparative Evaluation of Program Comprehension Measures.” *The Journal of Systems and Software* 52 (3): 121): 129.

Ericsson, K. A., and H. A. Simon. 1984. *Protocol Analysis: Verbal Reports As Data*. Cambridge, MA: The MIT Press.

Fincham, J. M., J. R. Anderson, S. Betts, and J. Ferris. 2010. “Using Neural Imaging and Cognitive Modeling to Infer Mental States while Using an Intelligent Tutoring System.” In *Proceedings of the 3rd International Conference on Educational Data Mining*, 51–60.

Freyd, M. 1923. “The Graphic Rating Scale.” *Journal of Educational Psychology* 14 (2): 83.

Fritz, T., A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger. 2014. “Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development.” In *Proceedings of the 36th International Conference on Software Engineering*, 402–413.

Galán, F. C., and C. R. Beal. 2012. “EEG Estimates of Engagement and Cognitive Workload Predict Math Problem Solving Outcomes.” In *International Conference on User Modeling, Adaptation, and Personalization*, 51–62.

Gevins, A., M. E. Smith, L. McEvoy, and D. Yu. 1997. “High-resolution EEG Mapping of Cortical Activation Related to Working Memory: Effects of Task Difficulty, Type of Processing, and Practice.” *Cerebral Cortex (New York, NY: 1991)* 7 (4): 374–385.

Gopstein, D., J. Iannacone, Y. Yan, L. DeLong, Y. Zhuang, M. K. C. Yeh, and J. Cappos. 2017. “Understanding Misunderstandings in Source Code.” In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 129–139.

Gopstein, D., H. H. Zhou, P. Frankl, and J. Cappos. 2018. “Prevalence of Confusing Code in Software Projects: Atoms of Confusion in the Wild.” In *Msr ’18: 15th International Conference on Mining Software Repositories, May 28–29, 2018, Gothenburg, Sweden*, 281–291.

Green, T. 1977. “Conditional Program Statements and Their Comprehensibility to Professional Programmers.” *Journal of Occupational Psychology* 50 (2): 93–109.

Gruber, T., A. Keil, and M. M. Müller. 2001. “Modulation of Induced Gamma Band Responses and Phase Synchrony in a Paired Associate Learning Task in the Human EEG.” *Neuroscience Letters* 316 (1): 29–32.

Haga, S., H. Shinoda, and M. Kokubun. 2002. “Effects of Task Difficulty and Time-on-task on Mental Workload.” *Japanese Psychological Research* 44 (3): 134–143.

Ikutani, Y., and H. Uwano. 2014. “Brain Activity Measurement During Program Comprehension With NIRS.” In *2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 1–6.

Jensen, O., J. Gelfand, J. Kounios, and J. E. Lisman. 2002. “Oscillations in the Alpha Band (9–12 Hz) Increase with Memory Load During Retention in a Short-term Memory Task.” *Cerebral Cortex* 12 (8): 877–882.

Jensen, O., and C. D. Tesche. 2002. “Frontal Theta Activity in Humans Increases with Memory Load in a Working Memory Task.” *European Journal of Neuroscience* 15 (8): 1395–1399.

Kelly, S. P., E. C. Lalor, R. B. Reilly, and J. J. Foxe. 2006. “Increases in Alpha Oscillatory Power Reflect an Active Retinotopic Mechanism for Distracter Suppression During Sustained Visuospatial Attention.” *Journal of Neurophysiology* 95 (6): 3844–3851.

Klimesch, W. 1999. “EEG Alpha and Theta Oscillations Reflect Cognitive and Memory Performance: a Review and Analysis.” *Brain Research Reviews* 29 (2–3): 169–195.

Klimesch, W., M. Doppelmayr, T. Pachinger, and H. Russegger. 1997. “Event-related Desynchronization in the Alpha Band and the Processing of Semantic Information.” *Cognitive Brain Research* 6 (2): 83–94.

Klimesch, W., M. Doppelmayr, H. Russegger, and T. Pachinger. 1996. “Theta Band Power in the Human Scalp EEG and the Encoding of New Information.” *Neuroreport* 7: 1235–1240.

Klimesch, W., M. Doppelmayr, J. Schwaiger, P. Auinger, and T. Winkler. 1999. “‘Paradoxical’ Alpha Synchronization in a Memory Task.” *Cognitive Brain Research* 7 (4): 493–501.

Klimesch, W., G. Pfurtscheller, W. Mohl, and H. Schimke. 1990. “Event-related Desynchronization, ERD-Mapping and

Hemispheric Differences for Words and Numbers." *International Journal of Psychophysiology* 8 (3): 297–308.

Kosti, M. V., K. Georgiadis, D. A. Adamos, N. Laskaris, D. Spinellis, and L. Angelis. 2018. "Towards an Affordable Brain Computer Interface for the Assessment of Programmers' Mental Workload." *International Journal of Human–Computer Studies* 115: 52–66.

Krause, C. M., A. H. Lang, M. Laine, M. Kuusisto, and B. Pörn. 1996. "Event-related." *EEG Desynchronization and Synchronization During An Auditory Memory Task. Electroencephalography and Clinical Neurophysiology* 98 (4): 319–326.

Krause, C. M., L. Sillanmäki, M. Koivisto, C. Saarela, A. Häggqvist, M. Laine, and H. Hämäläinen. 2000. "The Effects of Memory Load on Event-related EEG Desynchronization and Synchronization." *Clinical Neurophysiology* 111 (11): 2071–2078.

Letovsky, S. 1987. "Cognitive Processes in Program Comprehension." *Journal of Systems and Software* 7 (4): 325–339.

Lientz, B. P., E. B. Swanson, and G. E. Tompkins. 1978. "Characteristics of Application Software Maintenance." *Communications of the ACM* 21 (6): 466–471.

Lin, Y. T., C. C. Wu, T. Y. Hou, Y. C. Lin, F. Y. Yang, and C. H. Chang. 2016. "Tracking Students' Cognitive Processes During Program Debugging–An Eye-movement Approach." *IEEE Transactions on Education* 59 (3): 175–186.

McKeithen, K. B., J. S. Reitman, H. H. Rueter, and S. C. Hirtle. 1981. "Knowledge Organization and Skill Differences in Computer Programmers." *Cognitive Psychology* 13 (3): 307–325.

Melia, U., F. Clariá, M. Vallverdú, and P. Caminal. 2014. "Filtering and Thresholding the Analytic Signal Envelope in Order to Improve Peak and Spike Noise Reduction in EEG Signals." *Medical Engineering & Physics* 36 (4): 547–553.

Miller, G. A. 1956. "The Magical Number Seven, Plus Or Minus Two: Some Limits on Our Capacity for Processing Information." *Psychological Review* 63 (2): 81.

Peitek, N., J. Siegmund, S. Apel, C. Kästner, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann. 2018. "A Look into Programmers' Heads." *IEEE Transactions on Software Engineering* 46 (4): 442–462. doi:10.1109/TSE.2018.2863303.

Pennington, N. 1987a. "Comprehension Strategies in Programming." In *Empirical Studies of Programmers: Second Workshop*, 100–113.

Pennington, N. 1987b. "Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs." *Cognitive Psychology* 19 (3): 295–341.

Petsche, H., H. Pockberger, and P. Rappelsberger. 1984. "On the Search for the Sources of the Electroencephalogram." *Neuroscience* 11 (1): 1–27.

Pfurtscheller, G., and F. L. Da Silva. 1999. "Event-related EEG/MEG Synchronization and Desynchronization: Basic Principles." *Clinical Neurophysiology* 110 (11): 1842–1857.

Ray, W. J., and H. W. Cole. 1985. "EEG Alpha Activity Reflects Attentional Demands, and Beta Activity Reflects Emotional and Cognitive Processes." *Science (New York, NY)* 228 (4700): 750–752.

Rivera, S. M., A. Reiss, M. A. Eckert, and V. Menon. 2005. "Developmental Changes in Mental Arithmetic: Evidence for Increased Functional Specialization in the Left Inferior Parietal Cortex." *Cerebral Cortex* 15 (11): 1779–1790.

Romero, P. 2004. "Structural Knowledge and Language Notational Properties in Program Comprehension." In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*, 223–225.

Rostami, S., A. Shenfield, S. Sigurnjak, and O. Fakorede. 2015. "Evaluation of Mental Workload and Familiarity in Human Computer Interaction With Integrated Development Environments Using Single-Channel EEG." In *Proceeding of PPIG 2015-26th annual workshop*, 7–21.

Sammer, G., C. Blecker, H. Gebhardt, M. Bischoff, R. Stark, K. Morgen, and D. Vaitl. 2007. "Relationship Between Regional Hemodynamic Activity and Simultaneously Recorded EEG-theta Associated with Mental Arithmetic-induced Workload." *Human Brain Mapping* 28 (8): 793–803.

Schooler, J. W., and T. Y. Engstler-Schooler. 1990. "Verbal Overshadowing of Visual Memories: Some Things are Better Left Unsaid." *Cognitive Psychology* 22 (1): 36–71.

Schooler, J. W., S. Ohlsson, and K. Brooks. 1993. "Thoughts Beyond Words: When Language Overshadows Insight." *Journal of Experimental Psychology: General* 122 (2): 166.

Sergeant, J., R. Geuze, and W. Van Winsum. 1987. "Event-related Desynchronization and P300." *Psychophysiology* 24 (3): 272–277.

Shneiderman, B. 1977. "Measuring Computer Program Quality and Comprehension." *International Journal of Man-Machine Studies* 9 (4): 465–478.

Siegmund, J., N. Peitek, A. Brechmann, C. Parnin, and S. Apel. 2020. "Studying Programming in the Neuroage: Just a Crazy Idea?." *Communications of the ACM* 63 (6): 30–34.

Siegmund, J., N. Peitek, C. Parnin, S. Apel, J. Hofmeister, C. Kästner, A. Begel, A. Bethmann, and A. Brechmann. 2017. "Measuring Neural Efficiency of Program Comprehension." In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 140–150.

Soloway, E., and K. Ehrlich. 1984. "Empirical Studies of Programming Knowledge." *IEEE Transactions on Software Engineering* 10 (5): 595–609.

Sterman, M. B., C. A. Mann, D. A. Kaiser, and B. Y. Suyenobu. 1994. "Multiband Topographic EEG Analysis of a Simulated Visuomotor Aviation Task." *International Journal of Psychophysiology* 16 (1): 49–56.

Tadel, F., S. Baillet, J. C. Mosher, D. Pantazis, and R. M. Leahy. 2011. "Brainstorm: a User-friendly Application for MEG/EEG Analysis." *Computational Intelligence and Neuroscience 2011* 8.

Teasley, B. E. 1994. "The Effects of Naming Style and Expertise on Program Comprehension." *International Journal of Human-Computer Studies* 40 (5): 757–770.

Vogt, F., W. Klimesch, and M. Doppelmayr. 1998. "High-frequency Components in the Alpha Band and Memory Performance." *Journal of Clinical Neurophysiology* 15 (2): 167–172.

Weidenbeck, S. 1986. "Processes in Computer Program Comprehension." In *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*, 48–57.

Wiedenbeck, S., and V. Ramalingam. 1999. "Novice Comprehension of Small Programs Written in the Procedural and Object-oriented Styles." *International Journal of Human-Computer Studies* 51 (1): 71–87.

Wilson, T. D. 1994. “The Proper Protocol: Validity and Completeness of Verbal Reports.” *Psychological Science* 5 (5): 249–252.

Yamamoto, S., and S. Matsuoka. 1990. “Topographic EEG Study of Visual Display Terminal (VDT) Performance with Special Reference to Frontal Midline Theta Waves.” *Brain Topography* 2 (4): 257–267.

Yeh, M. K. C. 2018. “Examining Novice Programmers’ Software Design Strategies Through Verbal Protocol Analysis.” *International Journal of Engineering Education* 34 (2(A)): 458–470.

Appendix. Raw ERD Data

In this section, we present the data that we use to generate the results in Section 5.2.

Table A1 shows the ERD values of confusing (C) and non-confusing (NC) questions of each atom type in the alpha and theta frequency bands, and the *p*-values of Wilcoxon signed rank tests. This table helped to create Table 7, where ERD_{diff} values in Table 7 were created by taking the differences between the ERD values of non-confusing and confusing questions in this table.

Tables A2 and A3 helped to create Table 8. They show the ERD values of confusing (C) and non-confusing (NC) questions of each atom type, in the alpha and theta frequency bands, from the frontal and parietal regions, respectively. They also show the *p*-values of Wilcoxon signed rank tests. ERD_{diff} values in the frontal and parietal regions in Table 8 were created by taking the differences between the ERD values of non-confusing and confusing questions in Tables A2 and A3, respectively.

Table A1. ERD values of confusing (C) and non-confusing (NC) questions, and the *p* values of wilcoxon signed rank tests. An Asterisk (*) Indicates a Statistically Significant Difference (*p*<.05). This Table was Used to Create Table 7.

Atom Type	Alpha			Theta		
	C	NC	*p*	C	NC	*p*
Logic as Control Flow	−68.830204	−47.518116	0.0125*	−66.324641	−48.079281	0.0125*
Type Conversion	−26.854449	−56.789458	0.0218*	−14.198329	−55.035655	0.0166*
Preprocessor in Statement	−55.96725	−14.062117	0.0166*	−50.577432	−13.48642	0.0125*
Assignment as Value	2.2599107	−29.179533	0.1688	44.377624	−25.499828	0.0469*
Post-Increment/Decrement	−43.424223	−42.634	0.9594	−37.90009	−38.784971	0.7989
Change of Literal Encoding	7.4355928	16.747403	0.4446	0.24425831	21.027087	0.0593

Table A2. ERD values of confusing (C) and non-confusing (NC) Questions in the **Frontal** region, and the *p* values of wilcoxon signed rank tests. An Asterisk (*) Indicates a Statistically Significant Difference (*p*<.05). This Table was Used to Create Table 8.

Atom Type	Alpha			Theta		
	C	NC	*p*	C	NC	*p*
Logic as Control Flow	−69.28597	−47.41248	0.016605*	−66.92232	−49.39089	0.016605*
Type Conversion	−26.30449	−57.8885	0.028417*	−14.72777	−54.68286	0.028417*
Preprocessor in Statement	−58.1835	−15.34719	0.016605*	−51.38753	−13.61141	0.012515*
Assignment as Value	−1.226459	−29.92357	0.284503	45.47733	−27.61711	0.046853*
Post−Increment /Decrement	−43.12748	−39.86822	0.959354	−37.01905	−37.68987	0.798859
Change of Literal Encoding	7.464183	16.86986	0.575062	1.052563	20.01603	0.241121

Table A3. ERD values of confusing (C) and non-confusing (NC) questions in the **Parietal** region, and the *p* values of wilcoxon signed rank tests. An Asterisk (*) Indicates a Statistically Significant Difference (*p*<.05). This Table was Used to Create Table 8.

Atom Type	Alpha			Theta		
	C	NC	*p*	C	NC	*p*
Logic as Control Flow	−66.63633	−47.44212	0.010862*	−63.14127	−42.85918	0.020879*
Type Conversion	−36.06961	−52.93156	0.010862*	−20.21382	−57.42015	0.010862*
Preprocessor in Statement	−45.55478	−10.91749	0.138641	−46.52172	−14.51278	0.085831
Assignment as Value	11.24293	−26.52273	0.260393	34.71837	−16.21706	0.260393
Post−Increment/Decrement	−40.39156	−57.86189	0.173071	−37.04911	−47.41578	0.678402
Change of Literal Encoding	2.936996	7.989058	0.441268	−15.32108	10.15765	0.050612