A Framework that Explores the Cognitive Load of CS1 Assignments Using Pausing Behavior

Joshua Urry
joshua.urry@usu.edu
Computer Science
Utah State University
Logan, Utah, USA

John Edwards
john.edwards@usu.edu
Computer Science
Utah State University
Logan, Utah, USA

ABSTRACT

Pausing behavior in introductory Computer Science (CS1) courses has been related to course outcomes and could be linked to a student's cognitive load. Using Cognitive Load Theory and Vygotsky's Zone of Proximal Development as a theoretical framework, this study empirically analyzes keystroke latencies, or pause times between keystrokes, with the goal of better understanding what types of assignments need more scaffolding than others. We report the characteristics of eleven assignments, introduce a method to analyze pausing behavior, and investigate how pausing behavior changes with assignment characteristics (e.g., introducing new programming constructs, engaging creativity through Turtle graphics, etc). We find evidence that pausing behavior does change based on the assignment characteristics and that assignments with particular characteristics, such as object-oriented principles, may be more likely to have excessive demands on student working memory. We also find evidence that assignment completion time may not be an accurate measure of assignment difficulty.

CCS CONCEPTS

• **Social and professional topics** → **CS1**.

KEYWORDS

CS1, Keystrokes, Pauses, Engagement, Cognitive Load, Scaffolding

ACM Reference Format:
Joshua Urry and John Edwards. 2024. A Framework that Explores the Cognitive Load of CS1 Assignments Using Pausing Behavior. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630760

1 INTRODUCTION

There are multiple psychological theories that have helped guide and improve educational processes, such as Cognitive Load Theory and Vygotsky's Zone of Proximal Development. These theories have also been applied to Computing Education Research (CER) and may especially be applicable to introductory Computer Science (e.g., CS1) classes. Learning to program requires substantial working memory allocation. Instructors need to ensure that students have the necessary scaffolding to accomplish tasks that are just outside of their ability (i.e., the Zone of Proximal Development) so that their working memory is not overloaded, disabling learning and prompting disengagement. The concepts learned in CS1 courses will contribute to mental models of Computer Science that students will carry throughout their careers, so it is crucial for a CS1 instructor to be cognizant of assignments that cause students to feel overwhelmed and disengaged. An objective measure of pausing behavior would help instructors identify such assignments and provide the necessary scaffolding.

We have observed that it is a temptation for instructors (including ourselves) to judge the difficulty of an assignment by how long it takes for students to complete it. Doing this fails to take cognitive load and frustration into account. We suggest that it is important to find measures of difficulty that say meaningful things about student experience beyond time spent. Recently, there have been studies in CER that have examined the length of student's pauses between keystrokes (i.e., latencies), their effect on course outcomes, and their relation to a student's cognitive load [9, 20, 25]. We know of no studies that have incorporated latency analysis to compare assignments.

In this paper, we introduce a measure of assignment pausing behavior by analyzing student latencies in a CS1 course. We look at the proportion of pauses of different lengths, including short pauses (0 - 45 seconds), medium pauses (45 seconds – 6 minutes), and long pauses. Short pauses could represent a low cognitive load, while medium pauses could represent a high cognitive load. Long pauses are where a student is more likely to be disengaged [14], which could be caused by a working memory overload. We compare distributions of these pause lengths between assignments to discover what assignments may require more scaffolding. The research questions in this paper are:

RQ1: Are assignments different from each other in student pausing behavior?
RQ2: What types of assignments cause students to take the highest proportion of longer pauses?

The novelty of this research is that it compares the pausing behavior of students between assignments and provides instructors with a possible measure of cognitive load, as well as an indication of what assignments may require improved scaffolding. This knowledge will help instructors reduce the amount of overwhelmed and disengaged students and provide improved foundational instruction in Computer Science. Our contributions are an empirical framework for exploring Cognitive Load Theory and the Zone of Proximal Development in CS1 assignments, as well as an alternative

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630760

way of describing assignments based on student pausing behavior during assignment completion.

The findings of this study are that assignments differ in their pausing behavior. Furthermore, assignments that require CS1 students to implement object-oriented programming constructs have the highest proportion of longer pauses, which could indicate these types of assignments cause students to experience a working memory overload and may need additional scaffolding. Although, further research is needed to validate the cause of longer pauses in these assignments. In any case, we find evidence that assignments where students take longer pauses, which may signify a high cognitive load, do not necessarily take students the longest to complete.

2 RELATED WORK

2.1 Cognitive Load Theory and Zone of Proximal Development

Our work is based on two prominent psychological theories, Cognitive Load Theory and the Zone of Proximal Development (ZPD). Cognitive Load Theory suggests that the cognitive load (e.g., working memory allocation) required to learn material differs if the task is a biological primary or biological secondary task [13]. Biological primary tasks include skills that are necessary for survival, such as learning to speak a native language, and learning happens without much cognitive effort. Biological secondary skills have not historically been necessary for survival and require more cognitive effort and working memory allocation from the learner [13, 28]. Furthermore, a learner only has so much room in their working memory. If they try to learn a task that has too high of a cognitive load, their working is overwhelmed, and learning is frustrated [28]. As such, Cognitive Load Theory is used to inform educational processes. Cognitive Load Theory has been applied to learning academic subjects in a second natural language [1, 23], as well as technology-assisted learning [29].

Cognitive Load Theory could apply especially to CER [3, 24], where students learn new programming languages, interact with technology, and apply mathematical concepts. All of these are biological secondary skills. Learning a programming language is like learning a second natural language because students need to both learn the syntax of a programming language and the constructs behind the syntax. The syntax of a programming language has been referred to as "extraneous cognitive load" since it is necessary for programming, but not usually the main focus of computer science education [21]. Edwards et al. found that teaching students the syntax of a programming language before the problem-solving aspects of programming (e.g., a "syntax-first" pedagogy) leads to improved course outcomes. This is because students are freed of the "extraneous cognitive load" of the syntax and have more room in working memory for the higher-level constructs of computer programming [6, 10]. Studies of visual/block-based languages, with which syntactic errors are not possible, have similar findings [34].

The Zone of Proximal Development (ZPD) is a concept in educational psychology that was created by Lev Vygotsky [32]. It refers to the area where a learner cannot accomplish a task alone and needs the guidance (e.g., scaffolding) of a more experienced peer or educator. This zone is where learning occurs [32]. The ZPD, particularly the concept of scaffolding, has been commonly used as a framework for general education [33], as well as in computer science education [2, 31]. The relation between scaffolding and cognitive load in CER was shown when Stachel et al. found that students who used a scaffolding tool in a CS1 lab assignment had lower levels of self-reported cognitive load, and received higher grades [26]. Overall, it is crucial for CS1 assignments to have the optimal amount of scaffolding to prevent a student from having a cognitive overload.

2.2 Pausing behavior

In this research, cognitive load and ZPD are explored by analyzing students' pausing behavior in CS1 assignments. Pausing behavior has been analyzed in multiple contexts outside of CER [18, 22]. In CER, keystroke analysis has been shown to be a useful tool in analyzing students' programming processes, as it provides more information than an assignment submission [30]. Some of the important data provided by keystroke analysis are keystroke latencies (i.e., the elapsed time between keystrokes). Keystroke latencies have been used to examine pausing behavior in CER in multiple studies [9, 19, 20, 25]. Leinonen et al. found that keystroke latency patterns could help create tailored learning experiences for students (e.g., scaffolding to keep students in the ZPD) [19]. Many of the studies incorporating latencies have examined their relation with course outcomes, and provide evidence that students who take a higher proportion of longer pauses tend to have worse course outcomes [9, 20, 25]. Leppänen et al. suggest that a higher number of pauses in programming may suggest a higher cognitive load because the student is unable to retain all of the necessary information in working memory and has to search other material [20]. Shrestha et al. found that students who pause more often tend to have worse course outcomes. Cluster analysis found two groups of students in regard to pausing behavior: one that takes fewer mid-to-long pauses and one that takes more [25].

Keystroke latencies have also been recently used to predict student engagement during programming assignments. Edwards et al. created a regression model based on student responses to prompts while programming to predict the probability that a student was on-task (e.g., engaged) based on the elapsed time since their last keystroke [8]. Hart et al. built on the regression model by incorporating a larger sample size, error analysis, and other techniques to provide more accurate results [14]. Overall, these studies provide evidence that pausing behavior does influence CS1 course outcomes and engagement. However, no known study has examined keystroke latency differences between assignments in CS1 courses.

2.3 CS1 Assignments

While there have not been any known studies examining the pausing behavior between assignments in a CS1 course, there have been multiple studies examining what makes a good programming assignment, or how to improve programming assignments [4, 11, 12, 17, 27, 35]. Layman et al. argued that programming assignments should be meaningful (i.e., they should relate to real-life problems) [17]. Stevenson and Wagner similarly suggested that students will work harder if an assignment involves real-world problems and solutions, focuses on topics from class, and is challenging and interesting [27]. Garcia noted that the presentation of

an assignment is important and that design patterns (e.g., context, assignment descriptions, and hints) provide scaffolding that allows students to independently learn [12]. Kussmaul also addressed scaffolding methods for multiple CS1 assignments [16]. Kinnunen and Simon examined how CS1 assignments affect some cognitive processes of students by studying how programming experiences during assignments affect a student's perceived self-efficacy [15]. Pausing behavior and assignment design have been shown to be important factors in a student's experience in a CS1 course. As such, it is important to examine both factors to learn if different constructs presented in assignments require more scaffolding. It is worth noting the dearth of research on CS1 assignment design principles that incorporate Cognitive Load Theory (see Section 2.1).

3 METHODS

3.1 Data

This study analyzes the differences in keystroke latencies between assignments in a CS1 course to determine which assignments may have the highest cognitive load and need more scaffolding. The programming language taught in the course was Python. We use a public keystroke dataset released in 2022 [7]. It was collected at Utah State University in 2021, deidentified, and made available for public use with the oversight of their IRB. The dataset contains over 2 million keystrokes from 44 students across 8 assignments. We excluded students who had less than 1000 keystrokes across all assignments from analyses and only examined "file edit" events. The final sample size was about 900,000 keystrokes from 43 students. The dataset also contains files for the academic information of students in the course, due dates for the assignments, and the assignment descriptions [5]. However, the keystroke and assignment description files are the only files used in this study. See Edwards et al. [7] for further details on the data and data collection process.

3.2 Assignment Breakout

Of the 8 programming assignments that had data collected, 2 were dropped from this study. One was the last assignment in the course, which only 23 of the 43 students completed. The instructor dropped one assignment from a student's grades, so students may have chosen to drop the last assignment. The second assignment that was dropped had a mean keystroke count of 999.3 with a standard deviation of 1679.3. We were unsure of the cause of the unusually high variability, so we dropped this assignment as well. Additionally, we dropped one student from a repeated-measures ANOVA because they only had three keystrokes for one of the assignments in the test.

Of the six remaining assignments, multiple included two to three tasks. Each task involved different requirements and sometimes more advanced programming than the other task(s) included in the same assignment. While programming constructs may build upon each other in subsequent tasks, each task required students to create new files and write new code. As such, we broke up the tasks and treated them as separate assignments. The rest of the paper will present data on 11 assignments. This allowed for more detailed analyses of what programming constructs affect student pausing behavior. Table 1 shows a summary of the programming concepts throughout assignments.

3.3 Latency binning

To compare the pausing behavior between assignments, we create three bins of elapsed time since the previous keystroke. Our bins are empirically based rather than equal frequency or equal interval. We refer to pauses of length 0 - 45 seconds (bin 1) as short pauses, those of length 45 seconds to 6 minutes (bin 2) as medium pauses, and over 6 minutes (bin 3) as long pauses.

We based our bin sizes on the regression model for student engagement in Hart et al.. Given the elapsed time since the previous keystroke, the model predicts the probability that a student is engaged, or on task. The model predicts that students have an 81% chance of being on task at 45 seconds and a 52% chance of being on task at 6 minutes. We use 45 seconds as the upper threshold of the first bin (short pauses) so that the first bin represents the proportion of time students are most likely on task and who may have a relatively small cognitive load. 6 minutes is used as the upper threshold of the second bin (medium pauses), which represents pauses where students have a chance of being disengaged but could also be taking a pause to review notes, search Stack Overflow, etc. This bin could represent a high cognitive load with a lower chance of a working memory overload. Finally, bin 3 (long pauses) represents pauses where students are more likely to be disengaged and experience a working memory overload.

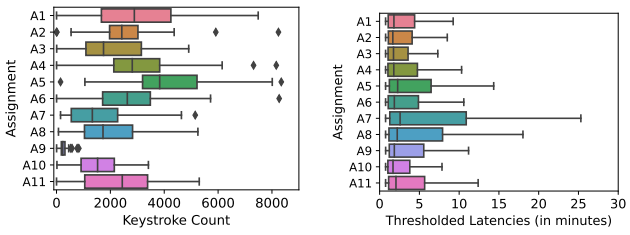
3.4 Statistical and visualization techniques

Since assignments vary in total keystrokes required for completion (see Figure 1a), comparing the raw keystroke counts in each pause length (i.e., short, medium, and long) would not yield an accurate comparison. It would just show how long an assignment took. For example, if one assignment had 3000 keystrokes with 10 long pauses and another assignment had 1500 keystrokes with 10 long pauses, a comparison of the raw counts of long pauses would show that these assignments were similar. However, it would be noteworthy that a student took just as many long pauses in an assignment that required only half the code. So, we normalize the pause lengths across each assignment per student by dividing the number of pauses in each bin by the total number of keystrokes. For example, if a student completed an assignment with 3000 short keystrokes, 100 medium keystrokes, and 10 long keystrokes, we would characterize their pausing behavior with a 3-dimensional vector $[\frac{3000}{3110}, \frac{100}{3110}, \frac{10}{3110}] = [0.965, 0.032, 0.003]$. This provides the proportion of latencies in each pause length.

After the raw counts are normalized, we scale the proportions for the parallel coordinate chart for visualization purposes, as over 90% of the latencies for every task are in the short pause bin. For each pause length in each assignment, we take the median of the proportions across submissions. Once the medians were collected for each assignment, we min-max scaled the values of each pause length between one and zero based on the other values in the pause length across assignments. As such, the assignment that had the highest proportion of pauses in a given pause length would be one, and the assignment that had the lowest proportion in a pause length would be zero. See Figure 2. Only the normalized values (before min-max scaling) were used in the ANOVA described in Section 4.3.

Assignment	Loops	Functions	Starter Code	Graphics	Modules	Classes	Lists	Searching	Sorting
A1	Y								
A2	Y								
A3	Y	Y							
A4	Y	Y	Y	Y					
A5	Y	Y	Y	Y	Y				
A6	Y	Y	Y	Y		Y			
A7	Y	Y	Y			Y			
A8	Y	Y	Y			Y	Y		
A9	Y	Y	Y				Y	Y	
A10	Y	Y	Y				Y	Y	Y
A11	Y	Y	Y			Y	Y	Y	Y

Table 1: Comparison of Task Characteristics



(a) Total keystrokes **(b) Thresholded latencies**

Figure 1: Boxplots of (a) total keystrokes per assignment and (b) latencies thresholded at 45 seconds per assignment.

4 RESULTS

4.1 Descriptive statistics

Figure 1a shows the total keystrokes across the eleven assignments. The total completion time distribution mirrors the keystroke distribution. Figure 1b shows latencies with a lower threshold of 45 seconds (e.g., latencies where students have a higher chance of being disengaged and could have a larger cognitive load). Assignment 7 has the most variability in thresholded pauses, as well as the highest median latency, which could signify that it had the highest cognitive load and potential for a working memory overload. The quartiles for total latencies (i.e., without a 45-second threshold) fall between 0 and 2 seconds for all assignments and show relatively little variation. Figure 1b) does not show outliers because there are many, due to the large number of keystrokes.

4.2 Parallel coordinates

Figures 2 and 3 show parallel coordinate charts that visualize the scaled short, medium, and long pauses for each assignment. Figure 2 compares the pausing behavior of Turtle graphics-based assignments against all other assignments. The Turtle graphic-based assignments (i.e., A4 – A6) share very similar latency patterns, even

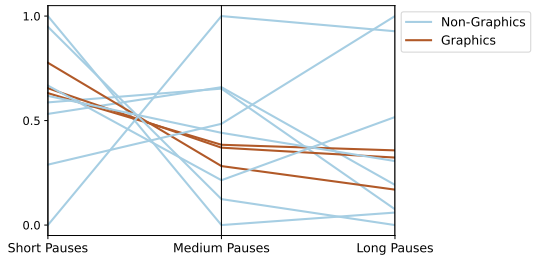


Figure 2: Parallel coordinates of graphics vs. non-graphics assignments.

though Assignment 5 took students more keystrokes and time to complete than the other graphic assignments (see Figure 1a).

4.3 Assignment groups and pause length comparison

We performed K-means clustering on the scaled vectors of pause length counts to discover if there were different groups of pausing behavior among assignments. We used the elbow method to select the number of clusters. This is accomplished by plotting the number of clusters against the model's Sum of Squares Error (SSE) and looking for a K value where the SSE sharply stops decreasing (e.g., the "elbow"). We chose K to be 3. The cluster centers are reported in Table 2 and the groups are visualized in Figure 3.

In Group 1, students are taking an average number of short and medium pauses, relative to other assignments, and occasionally take a long pause. Group 2 includes assignments 7 and 8. This group is characterized by the smallest proportions of short pauses and the largest proportions of long pauses, meaning students take a higher proportion of long pauses in these assignments, compared to the others. Group 3 includes assignments 2 and 10. This group has the highest proportions in short pauses and the lowest proportions in medium and long pauses, meaning students are not taking many medium or long pauses.

To test our first research question, which was whether assignments differed from each other in their pausing behavior, we ran a repeated measures ANOVA for each pause length between assignments 2, 4, and 8, which are representative of cluster Groups 3, 1, and 2, respectively. We used a repeated-measures ANOVA because the same sample was used for each of the three assignments in the test and the test distributions are normal (Table 3). Figure 4 shows the distributions of students' proportion of pauses in each of the pause lengths for each of the selected assignments. The test results are reported in Table 4. There was a statistically significant difference for each pause length, meaning that at least one assignment significantly differed from the others in every pause length.

5 DISCUSSION

5.1 Assignment pausing behavior differentiation

Our first research question is: *Are assignments different from each other in student pausing behavior*? Differences in pausing behavior can be seen in the parallel coordinates charts. In addition to these

Cluster	Asgmts	Short	Medium	Long
Group 1	1,3,4,5,6,9,11	0.638177	0.429329	0.277252
Group 2	7,8	0.144536	0.742248	0.963668
Group 3	2,10	0.974385	0.062096	0.029890

Table 2: K-means Cluster Centers for short pauses, medium pauses, and long pauses

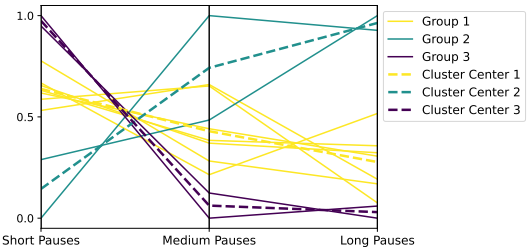


Figure 3: Parallel coordinates of K-means groups.

Asgmt	Short		Medium		Long	
	$\chi^2(2)$	P-value	$\chi^2(2)$	P-value	$\chi^2(2)$	P-value
A2	2.9	0.2	0.8	0.7	5.6	0.06
A4	2.3	0.3	2.5	0.3	2.6	0.3
A8	0.6	0.7	1.8	0.4	3.0	0.2

Table 3: D'Agostino-Pearson normality test results

Pause Length	F Value	P-value	η_p^2	90% CI
Short Pauses	75.9 (2, 52)	$3.8e^{-16}$	0.7	[0.6, 0.8]
Medium Pauses	44.7 (2, 52)	$5.2e^{-12}$	0.6	[0.5, 0.7]
Long Pauses	42.9 (2, 52)	$1.0e^{-11}$	0.6	[0.5, 0.7]

Table 4: Repeated-measures ANOVA results

visual measures, K-means clustering and the repeated-measures ANOVA also provide evidence that assignments in CS1 are different in their pausing behavior. This could suggest that different assignments cause different cognitive loads.

As reported in Section 4.2, Figure 2 shows that all of the Turtle graphics assignments (i.e., assignments 4, 5, and 6) share a strikingly similar pattern of scaled pauses across the three pause lengths. These assignments involved more creativity and less use of unfamiliar programming language constructs. This is seen in their pausing behavior. They all have a relatively high proportion of short pauses and middle to lower proportions of medium and long pauses. This signifies that, relative to other assignments, students tend to spend more time engaged with the assignment and do not have as high of a chance of being disengaged or experiencing a working memory overload. This similarity in pausing behavior is noteworthy, given that assignment 5 has the highest median keystroke count and total completion time out of all assignments (Figure 1a). Since this assignment took students a long time to complete, one might assume that it also had a high cognitive load. However, assignment 5 still shares similar pausing behavior to assignments 4 and 6, which have considerably lower median keystroke counts and time spent.

In addition to visually separating the data through parallel coordinates, K-means clustering found three groups of assignments that differ in their pausing behavior (Figure 3 and Table 2). The first group included most assignments evaluated in this study. Students took fewer breaks in these assignments. This group could signify the "normal" cognitive load for a CS1 assignment. Relative to other assignments, students spend most of their time engaged with their assignment, but may occasionally take a break in the medium or long pause range. Group 2 includes assignments 7 and 8, which could have the highest cognitive load. This group's cluster centers show students take a higher proportion of medium and long pauses when compared to other assignments. The characteristics of this group are discussed in detail in the next subsection, but both assignments require students to implement object-oriented programming more than other assignments. The number of scaled long pauses in this group suggests that these assignments have more pauses where students are disengaged than all other assignments. This could be caused by a working memory overload. Because of this, it is possible that these assignments did not provide enough scaffolding to keep students in the ZPD. The third group discovered by K-means also contains two assignments, assignments 2 and 10. These assignments have the highest scaled value of short pauses and the lowest value of medium and long pauses. These assignments could have a light cognitive load and are where students are the least likely to become disengaged. Neither of these assignments introduced anything new, so it is possible that students did not have to spend a lot of time thinking about the assignment. The three groups discovered by k-means clustering have clear visual distinctions in pausing behavior.

The final test to tell if there was a differentiation between pausing behavior in assignments was a repeated-measures ANOVA between assignments 2, 4, and 8 (i.e., groups 3, 1, and 2, respectively). This test found a difference across all pause lengths between these assignments. This test statistically confirmed that CS1 assignments do differ from each other in their pausing behavior.

5.2 Assignments that cause long pauses

Our next research question is *What types of assignments cause students to take the highest proportion of longer pauses?* Assignments 7 and 8 (i.e., group 2) have the highest scaled value of long pauses (e.g., the cluster center is at .97). Assignment 8 also has the highest value of medium pauses. The higher occurrence of medium and long pauses in these assignments could signify an increase in students' overall cognitive load, as suggested by Lee et al [18]. The higher number of medium pauses could be caused by a lack of room in a student's working memory, which prohibits them from storing the necessary information to continue with the assignment, forcing them to search other material [20]. Additionally, the relatively extreme proportion of long pauses signifies that students could be spending more time being disengaged in these assignments [14], which could be caused by a working memory overload and frustration, which stops learning [28].

Assignments 7 and 8 are similar in many ways. Both assignments involve creating a virtual environment where users enter input into

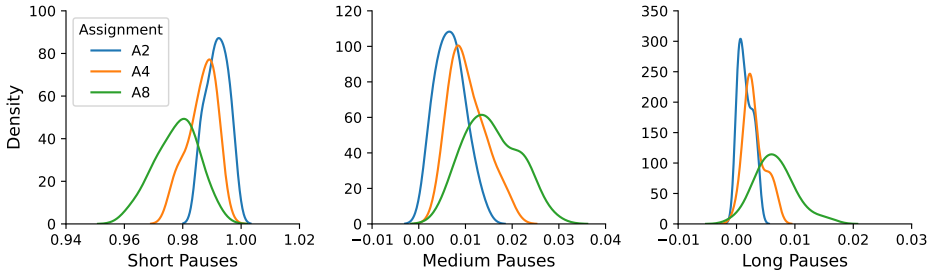


Figure 4: Assignment pause length comparisons.

a program to interact with virtual pets/beings (e.g., users can feed the pet, or ask to see the dimensions of a virtual being). The students must implement object-oriented programming methods to complete these assignments by creating classes for the pet/being that include multiple modules and functions. Students also need to consider how user input will affect the various aspects of the class, such as a timer that keeps track of the "age" of the pet/being. Other aspects of object-oriented programming, such as private data or operator overloading, are applied in these assignments. Students create a "main" file to tie together all of their other code in a program that asks for input from the user, who can interact with the pet/beings until they quit.

Assignments 6 and 11 also implement classes but do not have as high proportions of medium and long pauses. Assignment 6 included starter code that took care of the structure of the class. Students just had to fill in the Turtle graphics code. Assignment 11 also had starter code provided, but students still had to create some of the structure of the class. Since assignment 11 does not have as many long pauses as assignment 7 or 8, it is possible that it was not as intricate, or students had sufficient practice with classes by this assignment. Both of the descriptions for the assignments in group 3 (i.e., assignments 7 and 8) specifically mention that the assignment involves multiple tasks or problems, and one of the keys to completing the assignment is to break the assignment down into smaller pieces. These assignments have starter code and detailed assignment descriptions to help provide scaffolding to the students as they create the different classes. However, given the pausing behavior shown in our analyses, it is possible that the scaffolding is not enough and students fail to stay in the ZPD.

5.3 Threats to validity

There are several threats to the validity of this study. One is the relatively small sample size. The data is only from one class, that was taught in Python. It is possible that different instructors, assignment setups, and programming languages would yield different pausing behavior across constructs. It is also possible that the increased pausing behavior in object-oriented programming assignments is caused by other factors, such as students looking through starter code or switching between files, instead of a cognitive overload. Further research is needed to validate our findings. Additionally, we are basing our assumptions of cognitive load on pauses, which O'brien suggested are difficult to relate to cognitive load on their own [22]. While this was in another context, and more recent studies have suggested a relation between keystroke latencies and cognitive load [20], it could be beneficial to include another measure of cognitive load, such as a student's self-reported cognitive load, as was implemented in by Stachel et al., to corroborate the latency analysis [26]. Thirdly, while breaking out the tasks in each assignment did provide a more detailed view of pausing behavior by assignment construct, some tasks did have the same due date. The students' time management of when they completed all the tasks could have led some to be fatigued when completing the final task of the whole assignment, which may have affected their pausing behavior. Finally, our comparisons of assignment pausing behavior are all relative. We scale the pause length counts based on the other assignment pause length counts. It is possible that the addition of different assignments would change the scaled values.

6 CONCLUSIONS

This study has presented an empirical framework for using student pausing behavior to describe the possible cognitive load of eleven assignments in a CS1 course indicating which assignments could need more scaffolding. We find strong evidence that pausing behavior varies from assignment to assignment, suggesting that cognitive load also varies across assignments. This may seem obvious, but no study of which we are aware has quantitatively shown such differences in assignments. We find no evidence that pausing behavior is correlated with how long an assignment takes to complete, suggesting that length is an ineffective measure of difficulty. We also find that object-oriented principles may have caused students to pause more often, indicating, at least in the context of the dataset we used, that object-oriented assignments may need more scaffolding to keep assignments in the Zone of Proximal Development for most students.

Our future work will include a questionnaire where students can self-report their cognitive load and perceived assignment difficulty, allowing us to investigate whether pausing behavior can predict perceived difficulty. We are also considering expanding studies to other courses and languages, allowing us to further generalize our results.

ACKNOWLEDGMENTS

This paper is based upon work supported in part by the National Science Foundation under award DUE-2315783.

REFERENCES

[1] Nawal A. F. 2018. Cognitive load theory in the context of second language academic writing. *Higher Education Pedagogies* 3, 1 (2018), 385–402.

[2] Nicole Anderson and Tim Gegg-Harrison. 2013. Learning computer science in the" comfort zone of proximal development". In *Proceeding of the 44th ACM technical symposium on Computer science education*. 495–500.

[3] João Henrique Berssanette and Antonio Carlos de Francisco. 2021. Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review. *IEEE Transactions on Education* 65, 3 (2021), 440–449.

[4] Angela Carbone, John Hurst, Ian Mitchell, and Dick Gunstone. 2001. Characteristics of programming exercises that lead to poor learning tendencies: Part II. *ACM SIGCSE Bulletin* 33, 3 (2001), 93–96.

[5] John Edwards. 2022. 2021 CS1 Keystroke Data. https://doi.org/10.7910/DVN/BVOF7S

[6] John Edwards, Joseph Ditton, Dragan Trninic, Hillary Swanson, Shelsey Sullivan, and Chad Mano. 2020. Syntax exercises in CS1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 216–226.

[7] John Edwards, Kaden Hart, Raj Shrestha, et al. 2023. Review of CSEDM Data and Introduction of Two Public CS1 Keystroke Datasets. *Journal of Educational Data Mining* 15, 1 (2023), 1–31.

[8] John Edwards, Kaden Hart, and Christopher Warren. 2022. A Practical Model of Student Engagement While Programming. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*. 558–564.

[9] John Edwards, Juho Leinonen, and Arto Hellas. 2020. A study of keystroke data in two contexts: Written language and programming language influence predictability of learning outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 413–419.

[10] John M Edwards, Erika K Fulton, Jonathan D Holmes, Joseph L Valentin, David V Beard, and Kevin R Parker. 2018. Separation of syntax and problem solving in Introductory Computer Programming. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–5.

[11] Todd J Feldman and Julie D Zelenski. 1996. The quest for excellence in designing CS1/CS2 assignments. *ACM SIGCSE Bulletin* 28, 1 (1996), 319–323.

[12] Rita Garcia. 2021. Assignment Presentation Framework for CS1 Programming Problems. In *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[13] David C Geary and Daniel B Berch. 2016. Evolution and children's cognitive and academic development. *Evolutionary perspectives on child development and education* (2016), 217–249.

[14] Kaden Hart, John Edwards, and Christopher Warren. 2023. Accurate Estimation of Time-on-Task While Programming. In *In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 708–714.

[15] Päivi Kinnunen and Beth Simon. 2012. My program is ok–am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education* 22, 1 (2012), 1–28.

[16] Clifton L Kussmaul. 2008. Scaffolding for multiple assignment projects in CS1 and CS2. In *Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*. 873–876.

[17] Lucas Layman, Laurie Williams, and Kelli Slaten. 2007. Note to self: make assignments meaningful. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*. 459–463.

[18] Joy Yeonjoo Lee, Jeroen Donkers, Halszka Jarodzka, Géraldine Sellenraad, and Jeroen JG Van Merriënboer. 2020. Different effects of pausing on cognitive load in a medical simulation game. *Computers in Human behavior* 110 (2020), 106385.

[19] Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016. Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 132–137.

[20] Leo Leppänen, Juho Leinonen, and Arto Hellas. 2016. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 41–50.

[21] Raymond Lister. 2011. Computing education research programming, syntax and cognitive load. *ACM Inroads* 2, 2 (2011), 21–22.

[22] Sharon O'brien. 2006. Pauses as indicators of cognitive effort in post-editing machine translation output. *Across languages and cultures* 7, 1 (2006), 1–21.

[23] Stéphanie Roussel, Danielle Joulia, André Tricot, and John Sweller. 2017. Learning subject content through a foreign language should not ignore human cognitive architecture: A cognitive load theory approach. *Learning and Instruction* 52 (2017), 69–79.

[24] Philip Sands. 2019. Addressing cognitive load in the computer science classroom. *Acm Inroads* 10, 1 (2019), 44–51.

[25] Raj Shrestha, Juho Leinonen, Albina Zavgorodniaia, Arto Hellas, and John Edwards. 2022. Pausing While Programming: Insights From Keystroke Analysis. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 187–198.

[26] John Stachel, Daniela Marghitu, Taha Ben Brahim, Roderick Sims, Larry Reynolds, and Vernon Czelusniak. 2013. Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. *Journal of Integrated Design and Process Science* 17, 1 (2013), 37–54.

[27] Daniel E Stevenson and Paul J Wagner. 2006. Developing real-world programming assignments for CS1. *ACM SIGCSE Bulletin* 38, 3 (2006), 158–162.

[28] John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*. Vol. 55. Elsevier, 37–76.

[29] John Sweller. 2020. Cognitive load theory and educational technology. *Educational Technology Research and Development* 68, 1 (2020), 1–16.

[30] Arto Vihavainen, Matti Luukkainen, and Petri Ihantola. 2014. Analysis of source code snapshot granularity levels. In *Proceedings of the 15th annual conference on information technology education*. 21–26.

[31] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 117–122.

[32] Lev Semenovich Vygotsky and Michael Cole. 1978. *Mind in society: Development of higher psychological processes*. Harvard university press.

[33] Rob Wass and Clinton Golding. 2014. Sharpening a tool for teaching: the zone of proximal development. *Teaching in Higher Education* 19, 6 (2014), 671–684.

[34] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–25.

[35] Joanna Wolfe. 2004. Why the rhetoric of CS programming assignments matters. *Computer Science Education* 14, 2 (2004), 147–163.