5-2024

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

Joshua O. Urry
Utah State University, joshua.urry@usu.edu

Recommended Citation

Urry, Joshua O., "A Framework That Explores the Cognitive Load of CS1 Assignments Using Pausing Behavior" (2024). *All Graduate Theses and Dissertations, Fall 2023 to Present*. 145.
https://digitalcommons.usu.edu/etd2023/145

A FRAMEWORK THAT EXPLORES THE COGNITIVE LOAD OF CS1 ASSIGNMENTS USING PAUSING BEHAVIOR

by

Joshua O. Urry

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Data Science

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

2024

ABSTRACT

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

by

Joshua O. Urry, Master of Science

Utah State University, 2024

Major Professor: John Edwards, Ph.D.
Department: Computer Science

Pausing behavior in introductory Computer Science (CS1) courses has been related to course outcomes and could be linked to a student's cognitive load. Having an objective measure of the cognitive load would be beneficial to CS1 instructors as it would help them design assignments that are optimal for students' learning. Two studies are presented in this work. The first study uses Cognitive Load Theory and Vygotsky's Zone of Proximal Development as a theoretical framework and empirically analyzes keystroke latencies, or pause times between keystrokes, with the goal of better understanding what types of assignments need more scaffolding than others. The first study reports the characteristics of eleven assignments, introduces a method to analyze pausing behavior, and investigates how pausing behavior changes with assignment characteristics (e.g., introducing new programming constructs, engaging creativity through Turtle graphics, etc). Evidence is found 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. The findings also suggest that assignment completion time may not be an accurate measure of assignment difficulty. The second study builds on the first by validating whether the keystroke latency analysis corroborates results with a validated self-report measure of cognitive load. The latency analysis does have a relation to self-reported cognitive load, but not in the anticipated way. Short pauses have a positive linear relation with Intrinsic and Germane Cognitive load (ICL and GCL), while medium and long pauses have a negative linear relation. Furthermore, assignments that take students longer to complete do have higher ICL and GCL scores. Further research is needed to investigate the relation between keystroke latencies and cognitive load. However, both studies show assignment differentiation in pausing behavior, which suggests keystroke latency comparisons between assignments warrant more investigation.

(62 pages)

PUBLIC ABSTRACT

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

Joshua O. Urry

Pausing behavior in introductory Computer Science (CS1) courses has been related to a student's performance in the course and could be linked to a student's cognitive load, or assignment difficulty. Having an objective measure of the cognitive load would be beneficial to course instructors as it would help them design assignments that are not too difficult. Two studies are presented in this work. The first study uses Cognitive Load Theory and Vygotsky's Zone of Proximal Development as a theoretical framework to analyze pause times between keystrokes to better understand what types of assignments need more educational support than others. The first study reports the characteristics of eleven assignments, introduces a method to analyze pausing behavior, and investigates how pausing behavior changes with assignment characteristics (e.g., introducing new programming constructs, engaging creativity through Turtle graphics, etc). Evidence is found that pausing behavior does change based on the assignment type and that assignments with particular characteristics that cause students to break code into many small portions, and tie it all back together, may be more likely to have excessive demands on student working memory. Evidence is also found that assignment completion time may not be an accurate measure of assignment difficulty. The second study builds on the first study by validating whether the keystroke latency analysis corroborates results with a self-report questionnaire of cognitive load. The latency analysis does have a relation to self-reported cognitive load, but not in the manner anticipated. A higher proportion of shorter pauses is related to a higher cognitive load score, meaning students who spend more time typing have a higher cognitive load. Furthermore, assignments that take students longer to complete do have higher levels of cognitive load. Further research is needed to investigate the relation between keystroke latencies and cognitive load. However, both studies show assignment differentiation in pausing behavior between assignments, which suggests keystroke latency comparisons between assignments warrant more investigation.

To the family, faculty, and friends who helped me along the way.

ACKNOWLEDGMENTS

I would like to thank John, who spent hours with me on the experiment design, analysis, and scratching our heads about the results. I would also like to that Dr. Karimi and Dr. Hamdi for agreeing to be on my committee.

Joshua O. Urry

CONTENTS

Page

ABSTRACT . . . iii

PUBLIC ABSTRACT . . . v

ACKNOWLEDGMENTS . . . viii

LIST OF TABLES . . . xi

LIST OF FIGURES . . . xii

ACRONYMS . . . xiii

1 INTRODUCTION . . . 1

2 RELATED WORK . . . 5
- 2.1 Cognitive Load Theory and Zone of Proximal Development . . . 5
- 2.2 Pausing behavior . . . 6
- 2.3 Engagement during programming . . . 8
- 2.4 CS1 Assignments . . . 9

3 METHODS . . . 10
- 3.1 Study 1 . . . 10
  - 3.1.1 Data and experiment design . . . 10
  - 3.1.2 Assignment breakout . . . 10
  - 3.1.3 Latency binning . . . 11
  - 3.1.4 Statistical and visualization techniques . . . 12
- 3.2 Study 2 . . . 13
  - 3.2.1 Keystroke data . . . 13
  - 3.2.2 Questionnaire data and experiment design . . . 14

4 RESULTS . . . 16
- 4.1 Study 1 . . . 16
  - 4.1.1 Descriptive statistics . . . 16
  - 4.1.2 Parallel coordinates . . . 16
  - 4.1.3 Assignment groups and pause length comparison . . . 18
- 4.2 Study 2 . . . 21
  - 4.2.1 Descriptive statistics and parallel coordinates . . . 23
  - 4.2.2 Latencies vs. cognitive load . . . 24
  - 4.2.3 Completion time and starter code ratio vs. cognitive load . . . 30

5 DISCUSSION 32
5.1 Assignment pausing behavior differentiation 32
5.2 Assignments that cause long pauses 34
5.3 Pausing behavior vs. self-reported cognitive load 36
5.4 Assignment completion time vs. self-reported cognitive load 37
5.5 Threats to validity 38

6 CONCLUSION 40

REFERENCES 43

APPENDICES 47
A Cognitive Load Questionnaire 48

CURRICULUM VITAE 49

LIST OF TABLES

Table Page

3.1 Comparison of task characteristics in study 1 13

3.2 Comparison of task characteristics in study 2 14

4.1 Pause length medians and inter-quartile ranges for assignments in the first study. Sample is the number of submissions. 18

4.2 K-means Cluster Centers for short pauses, medium pauses, and long pauses 23

4.3 D'Agostino-Pearson normality test results 23

4.4 Repeated-measures ANOVA results 23

4.5 Assignment type, excerpt from the assignment description, description length in words, starter code length in characters, the average length of code submitted in characters, and the starter code to submitted code ratio for each assignment. 26

4.6 Pearson R correlation coefficients and P-values for each bin and load type . 28

LIST OF FIGURES

Figure Page

4.1 Boxplots of (a) total keystrokes per assignment in the first study and (b) total completion time for each assignment. 17

4.2 Boxplots of (a) all latencies per assignment in the first study in seconds and (b) latencies thresholded at 45 seconds per assignment. Outliers are not shown. 17

4.3 Parallel coordinates with assignment labels for the first study. 19

4.4 Parallel coordinates of graphics vs. non-graphics assignments in the first study. 19

4.5 Parallel coordinates of assignments with new vs known constructs in the first study. 20

4.6 Number of clusters (K) vs. SSE. 21

4.7 Parallel coordinates of K-means groups. 22

4.8 Assignment pause length comparisons. 22

4.9 Boxplots of (a) total keystrokes per assignment in the second study and (b) total completion time. 24

4.10 Boxplots of (a) all latencies in seconds and (b) latencies thresholded at 45 seconds in the second study. 25

4.11 Heatmap of cognitive load scores for each assignment by load type. 25

4.12 Parallel coordinates with assignment labels for the second study. 27

4.13 Parallel coordinates of graphics vs. game assignments in the second study. . 27

4.14 Parallel coordinates of object-oriented programming (OOP) tasks vs. tasks focusing on functions in the second study. 28

4.15 Scatterplot matrix of the median proportion of bin counts (columns) and the scores for each type of cognitive load (rows). 29

4.16 Scatterplot of completion time and load scores. 30

4.17 Scatterplot of starter code ratio and load scores. 31

ACRONYMS

CER	Computing Education Research
CS1	Introductory Computer Science Courses
ECL	Extraneous Cognitive Load
GCL	Germane Cognitive Load
ICL	Intrinsic Cognitive Load
OOP	Object-Oriented Programing
SSE	Sum of Squares Error
ZPD	Zone of Proximal Development

CHAPTER 1
INTRODUCTION

Multiple psychological theories 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 apply 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 a CS1 instructor must 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.

It is a temptation for instructors and students 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. I 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 (1; 2; 3). Given this, latency analysis could be used as an objective measure of a student's cognitive load. Much of the past research on this topic has focused on shorter pauses, such as milliseconds between keystrokes. There has been little research that has focused on longer latencies (e.g., minutes) when the probability of a student being disengaged is higher.

Additionally, no known studies have incorporated latency analysis to compare assignments in a CS1 course. There is a need to examine assignments through the lens of a theory, such as cognitive load theory, or the Zone of Proximal Development (ZPD), to guide the improvement of assignments. It is a common belief that programming assignments that take students a long time to complete are likely to have the most cognitive load and instances of disengagement. Operationalizing latency analysis to signify the cognitive load of an assignment could validate whether this is a correct assumption.

Two studies are reported in this paper. The first was previously published (4) and introduces a measure of assignment pausing behavior by analyzing student latencies in a CS1 course. It looks 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 (5), which could be caused by a working memory overload. The distributions of these pause lengths are compared between assignments to first, establish that student behavior, as measured by keystroke latency distribution, varies between assignments and second, to discover what assignments may require more scaffolding.

The second study acts as a follow-up to the findings of the first study by using a validated self-report questionnaire (6) to measure students' cognitive load. Keystroke data is collected as well, and the results of the two measures are compared to examine whether further evidence is found that the proposed framework using keystroke latencies could be used as an objective measure of cognitive load. The assignment completion time is also compared to the cognitive load scores to see if longer assignments cause more cognitive load. 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?

RQ3: Does student pausing behavior correlate with student self-reported cognitive load?

RQ4: Do assignments that have a longer completion time also have a higher cognitive load?

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. It also seeks to validate that measure. This knowledge will help instructors reduce the amount of overwhelmed and disengaged students and provide improved foundational instruction in Computer Science.

My contributions are:

1. An empirical framework for exploring Cognitive Load Theory and the Zone of Proximal Development in CS1 assignments.
2. A possible alternative way of describing assignments based on student pausing behavior during assignment completion.
3. Evidence that pausing behavior is related to cognitive load.

The findings of this study are that assignments differ in their pausing behavior. Furthermore, assignments that require CS1 students to implement programs where multiple files are tied together in a "main" file, such as object-oriented programming constructs (OOP), have the highest proportion of longer pauses. Additionally, the second study fails to provide evidence that the proposed latency analysis for pausing behavior correlates with the students' self-reported cognitive load in the way expected (i.e., more long pauses are related to more cognitive load). Instead, it is found that more short pauses are related to more cognitive load. So, while the relation was not in the expected direction, there is evidence that latency analysis could be a useful tool in analyzing cognitive load, though further research is needed to confirm this.

It is also found that, while assignments with longer pauses do not necessarily take longer for students to complete, students' cognitive load scores have a strong linear relation with assignment completion time. This means that they perceive that assignments that take a long time to complete are causing them more cognitive load. This could just be confirming the long-standing belief mentioned previously, that longer assignments are harder, or at least that they are perceived as being harder. Future work could incorporate a real-time measure of cognitive load to examine further if this is true. Overall, this thesis finds evidence that pausing behavior changes with assignment type and that latency analysis could provide an indication of cognitive load. These findings are beneficial to CS1 instructors as they could help them identify what assignments could be improved with more scaffolding.

CHAPTER 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 (7). 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 (7; 8). 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 (8).

Cognitive load has been traditionally broken out into three types: Intrinsic (ICL), Germane (GCL), and Extraneous (ECL) (6; 9; 10). ICL deals with the cognitive load of the material/task itself, while GCL encompasses the load required for the learner's construction of schemas, and ECL encompasses the load caused by the presentation of the material/task (9). All three types of cognitive load are important in learning, and there have been measures created that attempt to quantify each type of load (6). 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 (11; 12), as well as technology-assisted learning (13; 14).

Cognitive Load Theory could apply especially to CER (15; 16), 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 (17). 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 (18; 19). Studies of visual/block-based languages, with which syntactic errors are not possible, have similar findings (20).

The Zone of Proximal Development (ZPD) is a concept in educational psychology that was created by Lev Vygotsky (21). 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 (21). The ZPD, particularly the concept of scaffolding, has been commonly used as a framework for general education (22), as well as in computer science education (23; 24). 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 (25). 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 operationalized by analyzing students' pausing behavior in CS1 assignments. Pausing behavior has been analyzed in multiple contexts. For example, O'Brien analyzed pauses in post-editing machine translation and found that pauses vary across individuals, and do provide some indication of cognitive processing (26). In the context of academic writing, Révész et al. found evidence that the length of a pause in a writing task was related to the complexity of the task and the subsequent performance on the task (27). Furthermore, Borst et al. argued that the disruptiveness of an interruption depends on the complexity of the task, how long the disruption is, and when the interruption occurs (28). Lee et al. suggested that pausing increases a learner's overall cognitive load because the learner stimulates additional cognitive processes during a pause (29).

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). We use keystroke latencies as our measure of pausing behavior in this study. Keystroke latencies have been used to examine pausing behavior in CER in multiple studies (1; 2; 3; 31). Leinonen et al. found that keystroke latency patterns can show how much programming experience a student has, which could help create tailored learning experiences for students (31). A tailored learning experience could be thought of as the correct amount of scaffolding to keep a student in the ZPD. 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 (1; 2; 3). Leppänen et al. studied how students pausing behavior while typing, and how they spaced out their assignments, affected course outcomes. It was found that students who spaced out their work over multiple days tended to have higher exam scores compared to those who did not, while students who had more latencies that were a few minutes long tended to perform worse on exams. The researchers also 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 for other material (1).

Edwards et al. examined keystroke latencies across different programming languages and spoken languages. They found that small latencies (e.g., under 750 milliseconds) were mildly related to exam performance (2). Shrestha et al. conducted a study focused on CS1 student pausing behavior. Similar to previous studies, they found that students who pause more often tend to have worse course outcomes, and those who take more long pauses do worse than those who take a higher number of shorter pauses. Cluster analysis found two groups of students regarding pausing behavior. One that takes fewer mid-to-long pauses and one that takes more (3). Overall, these studies provide evidence that pausing behavior does influence CS1 course outcomes and could signify a higher cognitive load. However, no known study has looked at differences in pausing behavior between assignments in CS1 courses. This study will build on the previous research by examining differences in pausing behavior between assignments.

2.3 Engagement during programming

Keystroke latencies have also recently been used to predict student engagement during programming assignments. Edwards et al. used a PyCharm plugin to occasionally ask students if they were working on their assignment after a pause in keystrokes (32). Based on the data from the student responses, the researchers created a regression model to predict the probability that a student was on-task (e.g., engaged) based on the elapsed time since their last keystroke. Among other findings, it was shown that students worked on their assignments for about eight minutes before becoming disengaged (32).

Hart et al. built on the regression model by incorporating a larger sample size, error analysis, and other techniques to provide more reliable results (5). It was discovered that a threshold of five minutes could be used as a generalization for whether a student is engaged in a programming assignment, which can aid in calculating assignment completion time from a student's total keystroke latencies. Notably, using the enhanced regression model, students have an 81% chance of being on task after a 45-second latency and a 52% chance of being on task after a 6-minute latency (5). These will be used as thresholds for medium and long pauses, respectively, in our study.

Engagement, or time-on-task, has been shown to be an important factor in course outcomes (33), but we also incorporate it in the current study because of the implications for cognitive load and ZPD. Medium pauses, which still have a high probability of engagement, may indicate a high cognitive load without a working memory overload. Conversely, longer pauses, coupled with a lower probability of engagement, may indicate a working memory overload resulting in the loss of engagement. It is in these instances where more scaffolding may be necessary to help students stay in the ZPD.

2.4 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 (34; 35; 36; 37; 38; 39). Layman et al. argued that programming assignments should be meaningful (i.e., they should relate to real-life problems) (37). 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 (38). 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 (39). Kussmaul also addressed scaffolding methods for multiple CS1 assignments (40). 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 (41).

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. Pausing behavior differences in assignments could provide CS1 instructors with additional information on how to improve design patterns. Furthermore, it is worth noting the dearth of research on CS1 assignment design principles that incorporate cognitive load theory, which has been shown to be influential in education (see Section 2.1). While some studies do investigate scaffolding in CS1 assignments, there are still further applications for the ZPD in this context. The current study will add to the previous research on CS1 assignments by providing objective data to identify what assignments could cause a high cognitive load, or a working memory overload, rather than just making recommendations to improve assignments.

CHAPTER 3
METHODS

3.1 Study 1

3.1.1 Data and experiment design

The first study introduces the framework and 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. A public keystroke dataset is used that was released in 2022 (42). 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 (43). However, the keystroke and assignment description files are the only files used in this study. See Edwards et al. (42) for further details on the data and data collection process.

The general experiment design is as follows: First, we examined the assignment descriptions (included with the dataset (43)) and broke out the assignments by task. Then, we separated the latency counts for each assignment into three bins (i.e., short, medium, and long pauses). Finally, we applied statistical and visualization techniques to compare the pausing behavior between assignments.

3.1.2 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. I was unsure of the cause of the unusually high variability, so this assignment was dropped as well. Additionally, one student was dropped 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 requires 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 study 1 will present data on 11 assignments. This allowed for more detailed analyses of what programming constructs affect student pausing behavior. Table 3.1 shows a summary of the programming concepts throughout assignments.

3.1.3 Latency binning

To compare the pausing behavior between assignments, three bins of elapsed time since the previous keystroke are created. The bins are empirically based rather than equal frequency or equal interval. Pauses of length 0 - 45 seconds (bin 1) are referred to 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. I opted for these bins, rather than using equal interval width or equal frequency discretization, because of the practical significance. The majority of latencies for any assignment are less than a couple of seconds 4.10. This means that equal frequency bins would be seconds apart, with the last bin containing all of the medium-long pauses. This method would likely not be informative about the cognitive load, or just practically informative, for each assignment. Likewise, equal interval binning would be somewhat arbitrary.

The bin sizes are based on the regression model for student engagement by Hart et al.

(5). 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. I 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 pausing 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 may be more likely to be disengaged and experience a working memory overload.

3.1.4 Statistical and visualization techniques

Since assignments vary in total keystrokes required for completion (see Figure 4.1a), 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, the pause lengths are normalized 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, the median of the proportions across submissions is taken. Once the medians were collected for each assignment, I min-max scale 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 4.4. Only the normalized values (before min-max scaling) were used in the ANOVA described in Section 4.1.3.

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 3.1: Comparison of task characteristics in study 1

3.2 Study 2

3.2.1 Keystroke data

The second study uses the same latency analysis techniques described above. However, the data was collected in the Fall 2023 semester in one CS1 course at Utah State University with the oversight of the IRB. The dataset for the second study contained over 800,000 keystrokes from 35 students (i.e., after removing a study that had less than 1000 keystrokes and filtering for file events, as was done in the first study). Due to some issues with the tool used to collect keystroke data, only 5 assignments are included in this dataset. The last assignment was dropped from the study, as part of the assignment required students to search for bugs in the provided code and document what the bug was and how they fixed it. This causes the latencies for this assignment to be significantly skewed, as students would stop working in their IDE when they found a bug and document the findings elsewhere. This made it appear that students were taking very long breaks when they were still working on their assignments. Two of the remaining assignments have two tasks, so they are broken up into two separate assignments, as was done in the first study, which leaves 6 assignments for analysis. Table 3.2 shows the breakdown of task characteristics in the second study. The assignments in study two are shown as "T(task #)" to not be confused with the assignments from study 1.

Assignment	**Loops**	**Functions**	**Starter Code**	**Graphics**	**Modules**	**Classes**
T1	Y	Y	Y	Y		
T2	Y	Y	Y	Y	Y	
T3	Y	Y	Y			Y
T4	Y	Y	Y	Y		Y
T5	Y	Y	Y			Y
T6	Y	Y	Y	Y		Y

Table 3.2: Comparison of task characteristics in study 2

3.2.2 Questionnaire data and experiment design

The second study adds a self-report questionnaire modified from the validated questionnaire created by Klepsch et al. (6) that students filled out after submitting each assignment. See the appendix for a copy of the questionnaire A. The questionnaire contains 7 questions that separate Intrinsic, Germane, and Extraneous Cognitive Load. Responses are given on a 1 - 5 Likert scale (6). Each questionnaire was administered to students over Canvas after they submitted their assignments. If an assignment had two tasks, an additional question was added at the beginning of the questionnaire asking students which task they wanted to fill out the questionnaire for, as the proposed framework treats tasks as separate assignments. In total, 26 students regularly filled out each questionnaire. To further explore how the setup of each assignment affected the students' cognitive load scores, the average final code character length of the final submissions is gathered by first counting all of the "Insert" events in the keystroke data and subtracting the "Delete" and "Revert" events. These counts are then averaged by student. The ratio of starter code to final code characters is then calculated.

After the data is gathered, latency binning is carried out the same way as described in Study 1. Parallel coordinates charts are then created based on the values. The Cognitive Load questionnaire is then scored by first taking the average of the student's responses to the questions for each type of cognitive load (see A for the breakdown of which questions are Intrinsic, Germane, and Extraneous Cognitive load), then the student's responses were averaged per load type to produce an overall score for each assignment. The overall load score was then correlated with the median bin proportion for each pause length and each type of load to examine how they related. Additionally, the cognitive load scores are correlated with the median assignment completion time to test whether assignments that take longer are associated with more cognitive load.

CHAPTER 4
RESULTS

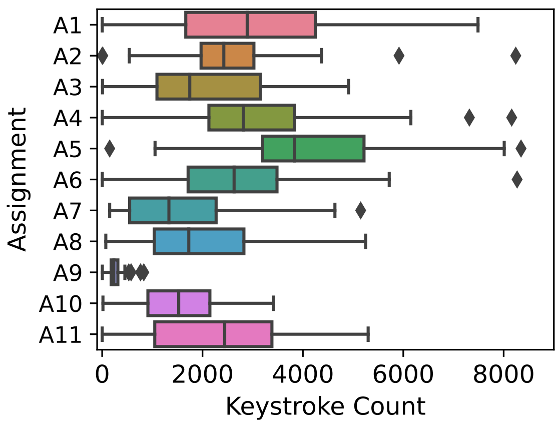
4.1 Study 1

4.1.1 Descriptive statistics

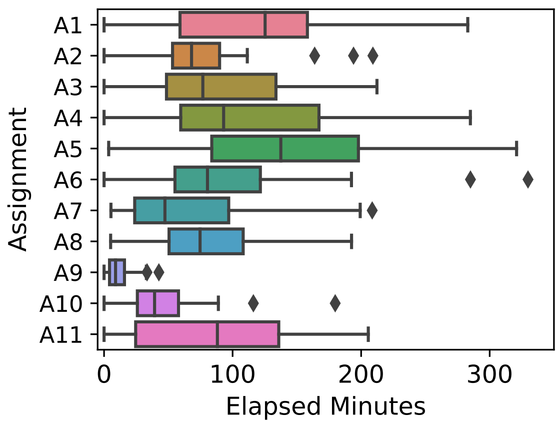
Figure 4.1a shows the total keystrokes across the eleven assignments, while Figure 4.1b shows the total time students took to complete each assignment. For the most part, these box plots match, but there are some minor differences (e.g., when comparing A6 - A8 in each chart). Figure 4.2a shows the breakdown of the total latencies (e.g., pauses) between assignments. The quartiles for these latencies fall between 0 and 2 seconds for all assignments and show relatively little variation between assignments. Figure 4.2b shows latencies with a lower threshold of 45 seconds (e.g., when students could have a higher chance of being disengaged and could have a larger cognitive load). Assignments notably have much more variability in this range. 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 boxplots of the latencies (i.e., Figures 4.2a and 4.2b) do not show outliers because there are many, due to the large number of keystrokes. Table 4.1 shows pause length medians and inter-quartile ranges of the latency counts in the short, medium, and long bins.

4.1.2 Parallel coordinates

Figures 4.3 - 4.7 show parallel coordinate charts that visualize the scaled short, medium, and long pauses for each assignment. Figure 4.3 shows the chart with assignment labels. Figures 4.4 and 4.5 compare the pausing behavior of different assignment characteristics. Figure 4.4 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

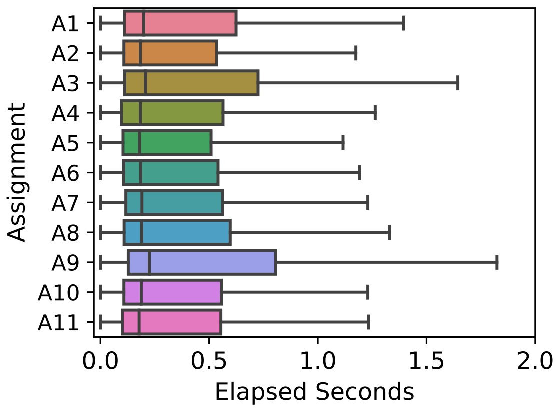


(a) Total Keystrokes

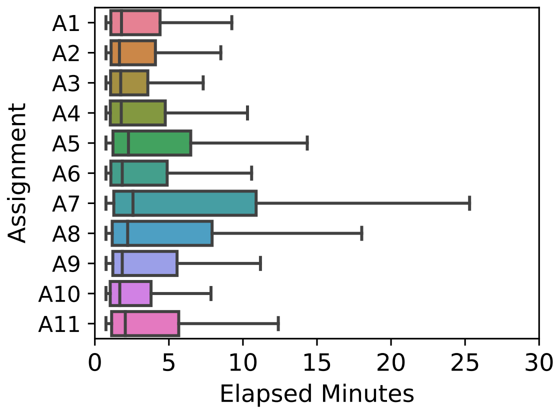


(b) Total Completion Time

Fig. 4.1: Boxplots of (a) total keystrokes per assignment in the first study and (b) total completion time for each assignment.



(a) All Latencies (sec)



(b) Thresholded latencies

Fig. 4.2: Boxplots of (a) all latencies per assignment in the first study in seconds and (b) latencies thresholded at 45 seconds per assignment. Outliers are not shown.

Assignment	Sample	Short		Medium		Long	
		Med	IQR	Med	IQR	Med	IQR
A1	42	2841.5	2495.3	35.5	32.8	7.0	8.0
A2	40	2391.0	1051.8	16.5	11.5	4.0	5.0
A3	33	1690.0	2002.0	21	32.0	3.0	4.0
A4	34	2788.0	1679.3	28.0	29.3	9.0	7.0
A5	34	3788.5	1959.5	38.5	38.8	13.0	13.8
A6	31	2584.0	1744.5	24.0	20.5	6.0	8.0
A7	32	1311.5	1690.8	14.5	20.3	9.5	7.3
A8	30	1681.0	1766.0	26.0	17.8	11.5	6.5
A9	31	229.0	131.5	2.0	4.5	1.0	1.0
A10	30	1510.5	1224.8	12.0	11.0	2.0	2.0
A11	28	2395.5	2259.8	25.5	37.3	7.5	7.8

Table 4.1: Pause length medians and inter-quartile ranges for assignments in the first study. Sample is the number of submissions.

latency patterns, even though Assignment 5 took students more keystrokes and time to complete than the other graphic assignments (see Figure 4.1a).

Figure 4.5 compares the pausing behavior of assignments where a new construct was introduced and assignments that did not introduce a new construct. For the most part, assignments that introduce a new construct fall in about the middle of most pause lengths with respect to their pausing behavior. This could be because most assignments that introduce a new construct tend to be simpler so the student can learn the construct. The exception is Assignment 8, which introduces lists. This assignment has the lowest proportion of short pauses, and some of the highest medium and high pauses, signifying that students took longer pauses for this assignment.

4.1.3 Assignment groups and pause length comparison

K-means clustering is performed on the scaled vectors of pause length counts to discover if there were different groups of pausing behavior among assignments. The elbow method is used 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"). 4.6 shows the plot. Though there were multiple small "elbows", K was set as three. I felt two clusters did not break out

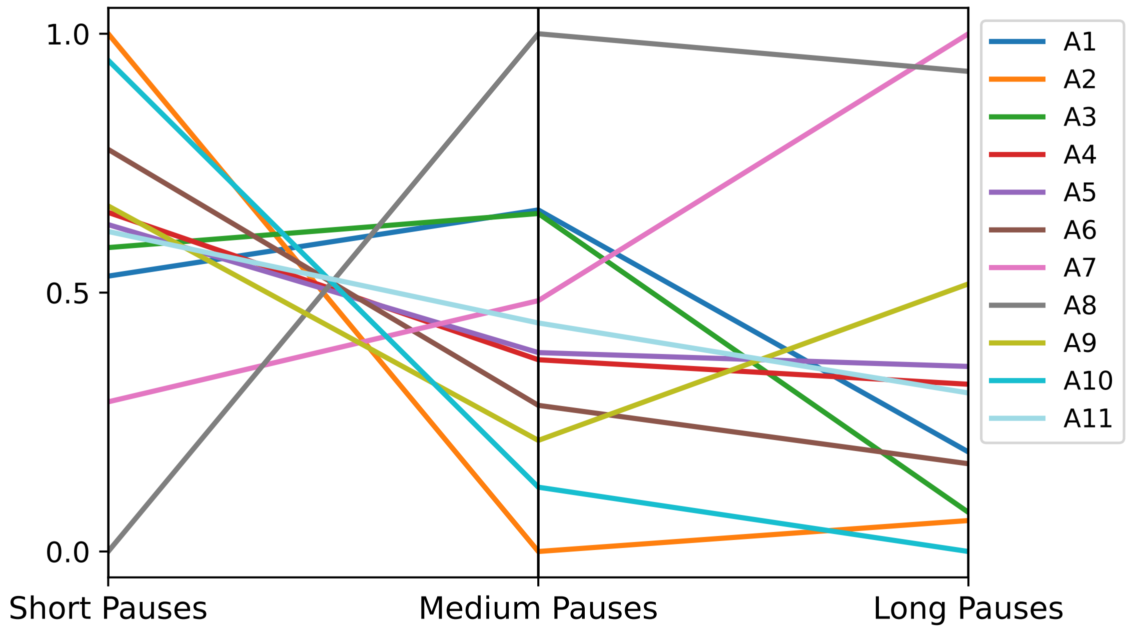


Fig. 4.3: Parallel coordinates with assignment labels for the first study.

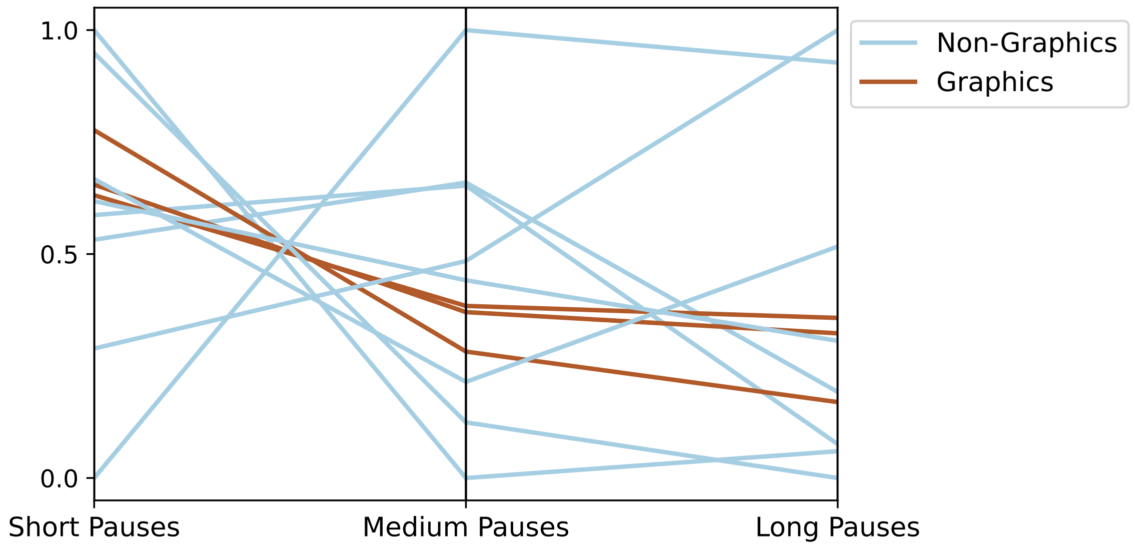


Fig. 4.4: Parallel coordinates of graphics vs. non-graphics assignments in the first study.

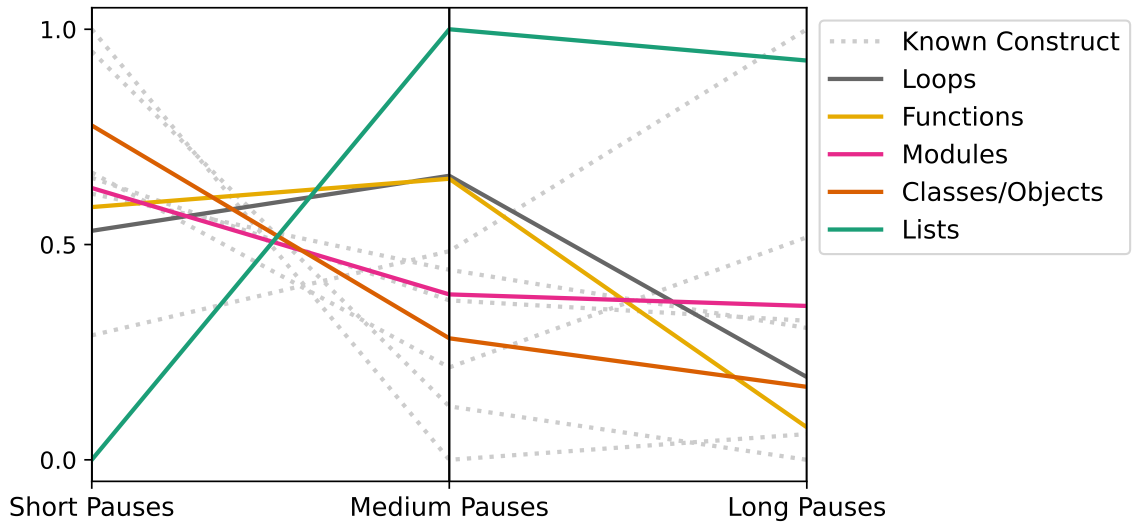


Fig. 4.5: Parallel coordinates of assignments with new vs known constructs in the first study.

the assignments enough, and more than three clusters produced groups that did not have enough discriminability. The cluster centers are reported in Table 4.2 and the groups are visualized in Figure 4.7.

Group 1 has short and medium pause proportions that are about in the middle of other assignments, and longer pauses on the lower end of the scaled values. As such, while they may have a higher cognitive load that requires students to take some medium pauses, they do not take many long pauses which could be caused by a working memory overload. This could be the optimal group for keeping students in the ZPD. 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. In other words, students are taking more pauses over 6 minutes in these assignments than in other assignments. The common attribute for assignment 7 and assignment 8 is that these assignments require students to implement object-oriented programming (e.g., classes, objects, etc.; OOP) to accomplish the assignment goals. Assignment 6 introduced classes, but the syntactical structure of the class was provided for the students in the starter code. Students just had to fill in the methods of the class with Turtle graphics code. Assignment 6 is a part of Group 1. Group

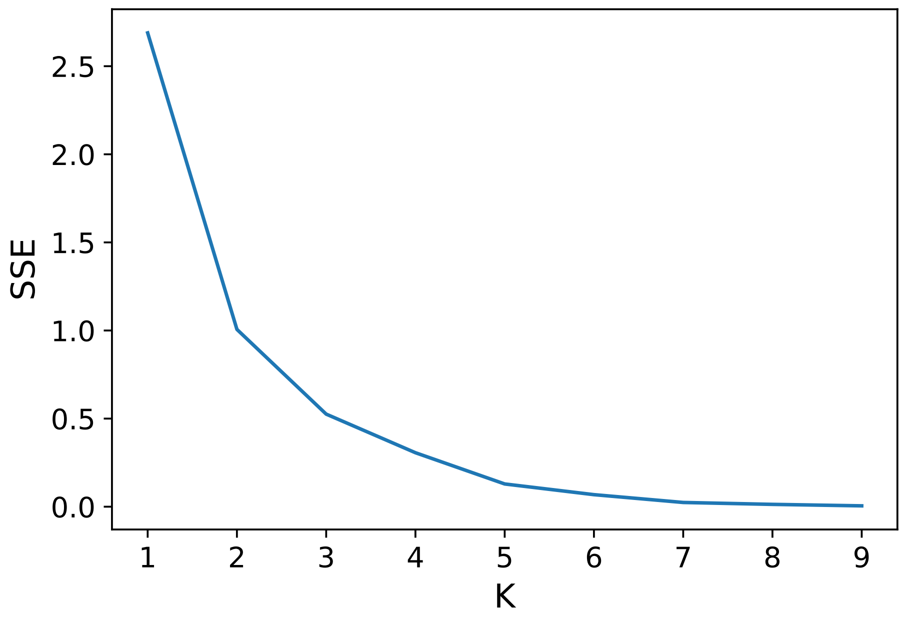


Fig. 4.6: Number of clusters (K) vs. SSE.

3 includes assignments 2 and 10. This group has the highest proportions in short pauses and the lowest proportions in medium and long pauses. Students do not take many breaks in these assignments, and it is possible that these assignments are not likely to cause a working memory overload.

To further test the first research question, which was whether assignments differed from each other in their pausing behavior, a repeated measures ANOVA was rand for each pause length between assignments 2, 4, and 8, which are representative of cluster Groups 3, 1, and 2, respectively. I 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 4.3). Figure 4.8 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 4.4. 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.

4.2 Study 2

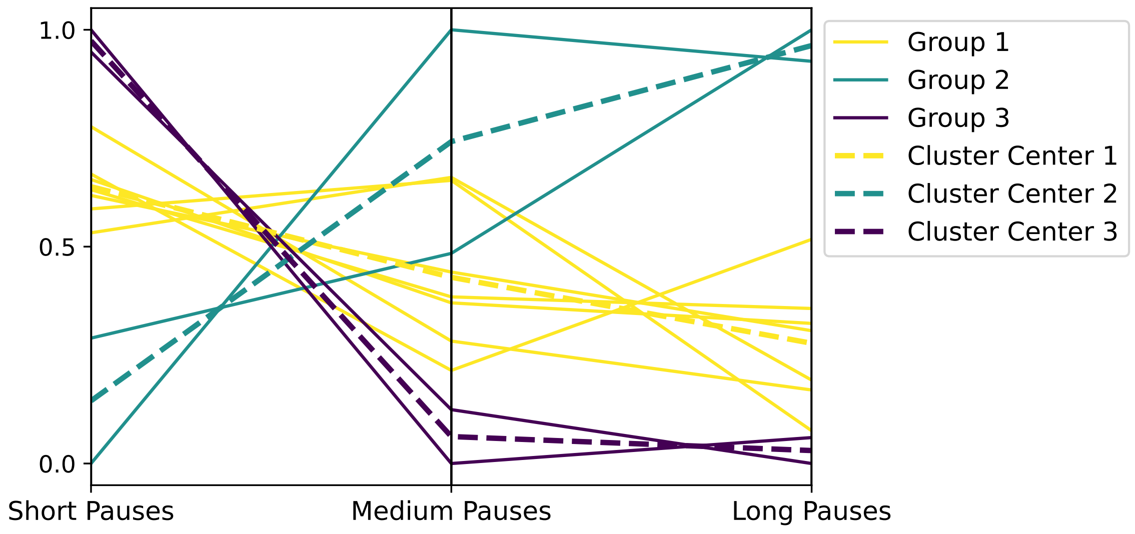


Fig. 4.7: Parallel coordinates of K-means groups.

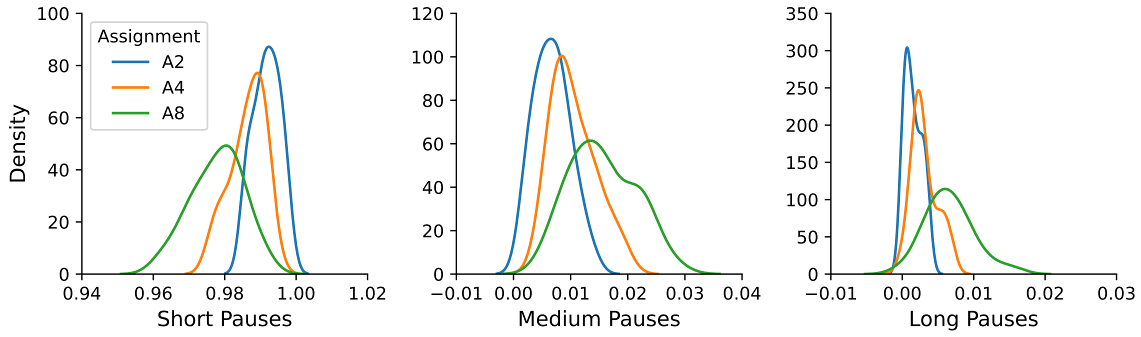


Fig. 4.8: Assignment pause length comparisons.

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

Assignment	**Short**		**Medium**		**Long**	
	χ^2**(2)**	**P-value**	χ^2**(2)**	**P-value**	χ^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

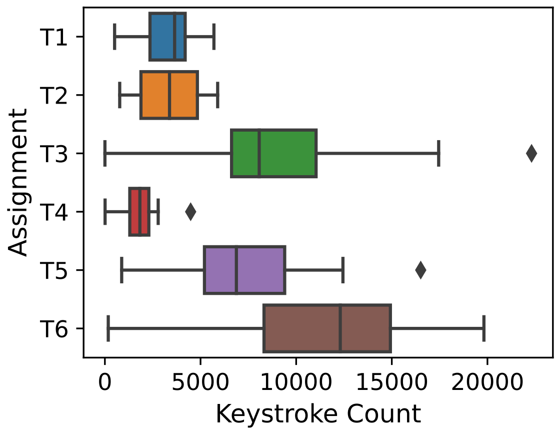
Table 4.3: D'Agostino-Pearson normality test results

4.2.1 Descriptive statistics and parallel coordinates

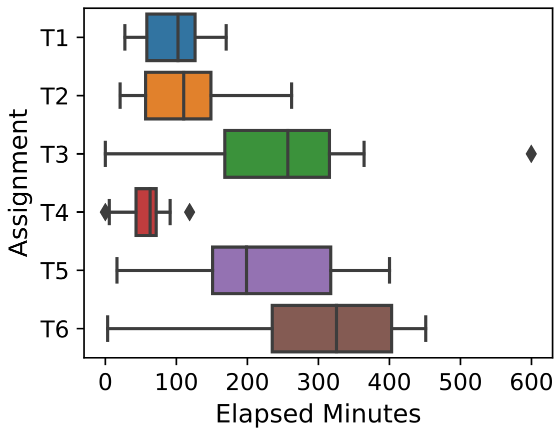
Similar to the first study, Figure 4.1a shows the total keystrokes across the 7 assignments, while Figure 4.9b shows the total completion time. Figure 4.10a shows the total latencies, while Figure 4.10b shows latencies thresholded at 45 seconds. Figure 4.11 shows the averaged scores for each type of cognitive load (i.e., Intrinsic, Germane, and Extraneous). Notably, assignments 3 and 5 have slightly higher scores across all load types. Additionally, Extraneous Load scores are lower across all assignments. Table 4.5 shows the type of assignment, a short excerpt for each assignment, the length of the assignment description, the length of the starter code, the average length of the student submissions, and the ratio of starter code to final submission. Figures 4.3 - 4.14 show variations of the parallel coordinates charts of bin proportions for the assignments in the second study. Figure 4.3 shows the chart with the individual assignment labels. Figure 4.13 shows the comparison of graphics vs game assignments in the second study (e.g., all assignments measured were either Turtle graphics or a game). Noticeably, graphics assignments appear to cause more

Pause Length	**F Value**	**P-value**	$\eta^2_{\mathbf{p}}$	**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 4.4: Repeated-measures ANOVA results



(a) Total Keystrokes (Study 2)



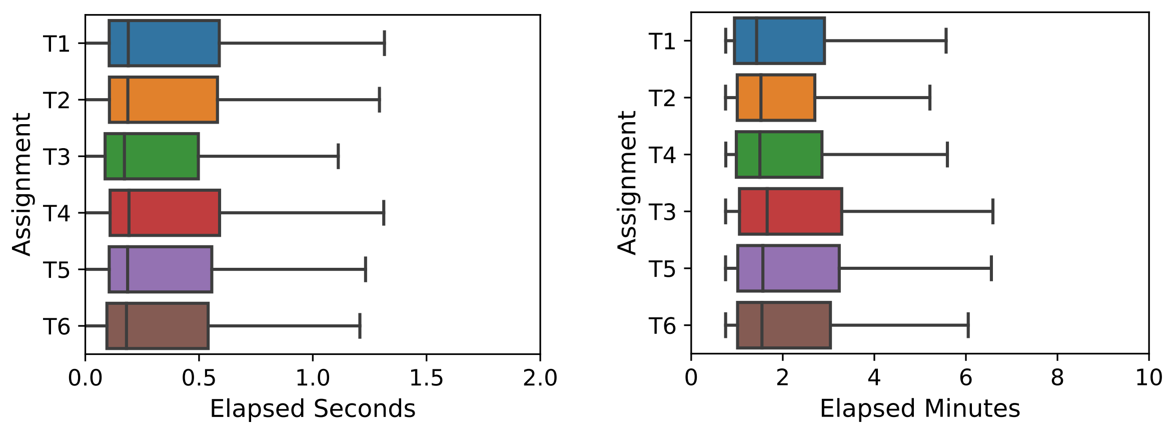
(b) Total Completion Time (Study 2)

Fig. 4.9: Boxplots of (a) total keystrokes per assignment in the second study and (b) total completion time.

medium and long pauses than games. Finally, Figure 4.14 shows the comparison of assignments that focused on OOP principles (e.g., classes, methods, etc.) compared to those that focused on functions. These were the main constructs in the assignments that were measured. OOP assignments had noticeable variability in the charts, but the assignment that caused the most medium pauses and second-most long pauses was an OOP assignment.

4.2.2 Latencies vs. cognitive load

To test whether the latency analysis in the framework developed in the first study was an accurate measure of cognitive load, the median proportion in each bin was correlated with the scores for each type of load. The scatterplot matrix in 4.15 shows scatterplots for each bin and load type combination. Notably, there is a positive correlation between the pause proportions in the short bin and negative correlations with the medium and long bins for Intrinsic and Germane Cognitive Load. This is contrary to the conclusions drawn in the first study. The Pearson R correlation coefficients, as well as the corresponding P-values, are given in Table 4.6. None of the tests were statistically significant, but the relation between long pauses and intrinsic load was trending significance.



(a) All Latencies (Study 2) (b) Thresholded latencies (Study 2)

Fig. 4.10: Boxplots of (a) all latencies in seconds and (b) latencies thresholded at 45 seconds in the second study.

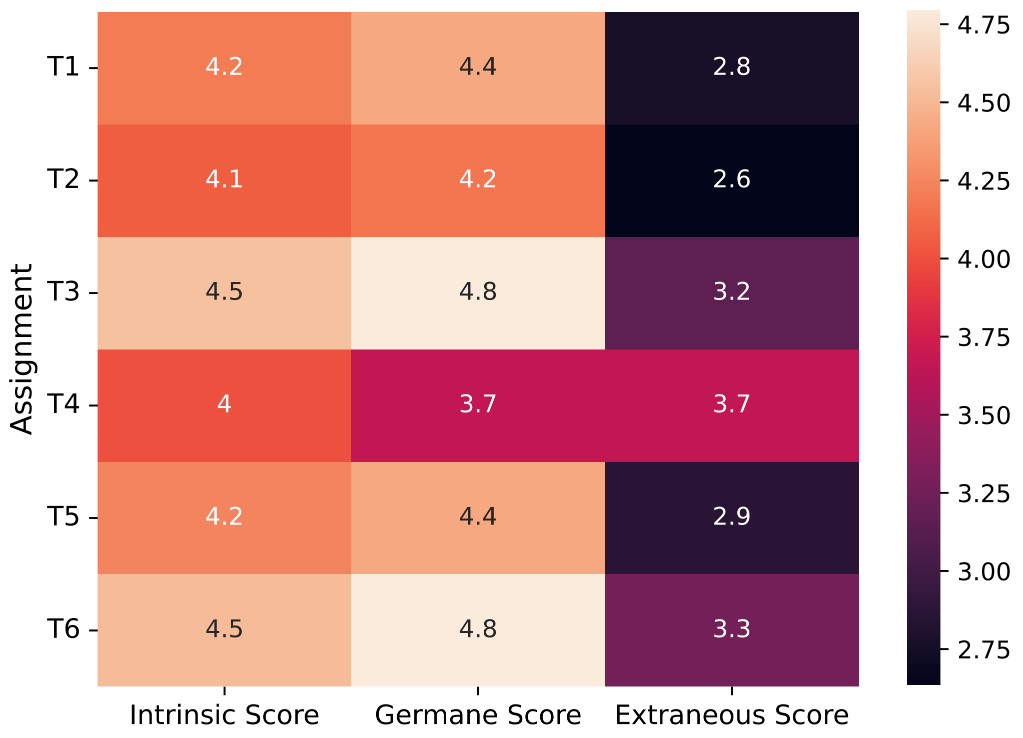


Fig. 4.11: Heatmap of cognitive load scores for each assignment by load type.

Assignment	Type	Description Excerpt	Description Length	Starter Length	Avg Submission Length	Starter:Submission	Short Pause Proportion
T1	Graphics	A program that prompts the user for some information and then draws a chessboard according to their specifications.	675	400	1753	0.21	0.9923
T2	Graphics	A program that takes user input, and then draws a pattern with different shapes and colors.	779	1379	1873	0.74	0.9909
T3	Game	A game where a character moves around to collect treasures all while avoiding a dangerous bomb. Students could choose the look and feel.	1126	1206	3863	0.31	0.9912
T4	Graphics	A class that works with the given starter code where a drawn face changes emotions based on a user click.	414	715	904	0.79	0.9902
T5	Game	A game where the player attempts to either catch or kill critters. The students can choose whatever critters and tools they would like, as long as the game follows the same technical premise.	1313	1224	3281	0.37	0.992
T6	Game	Students create a program that has them and a friend racing all over the city of "Crashtropolis". Speed and sharp turns are the name of the game. So is crashing into walls and getting zapped by lasers. The requirements for the program are looser than others.	1210	1159	5034	0.23	0.9937

Table 4.5: Assignment type, excerpt from the assignment description, description length in words, starter code length in characters, the average length of code submitted in characters, and the starter code to submitted code ratio for each assignment.

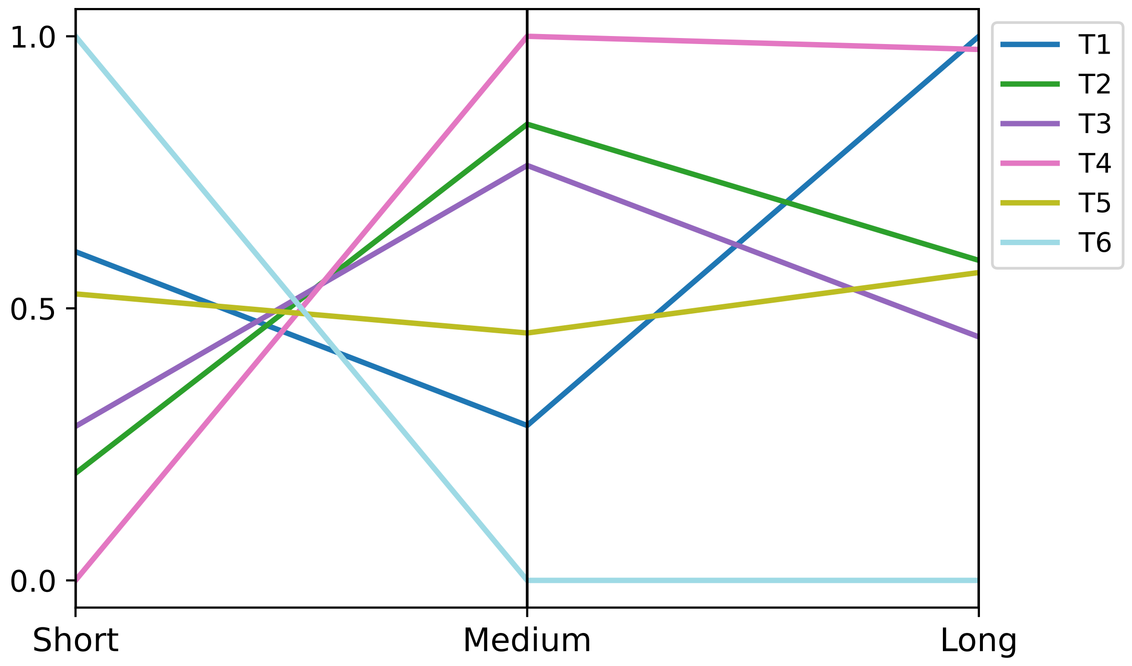


Fig. 4.12: Parallel coordinates with assignment labels for the second study.

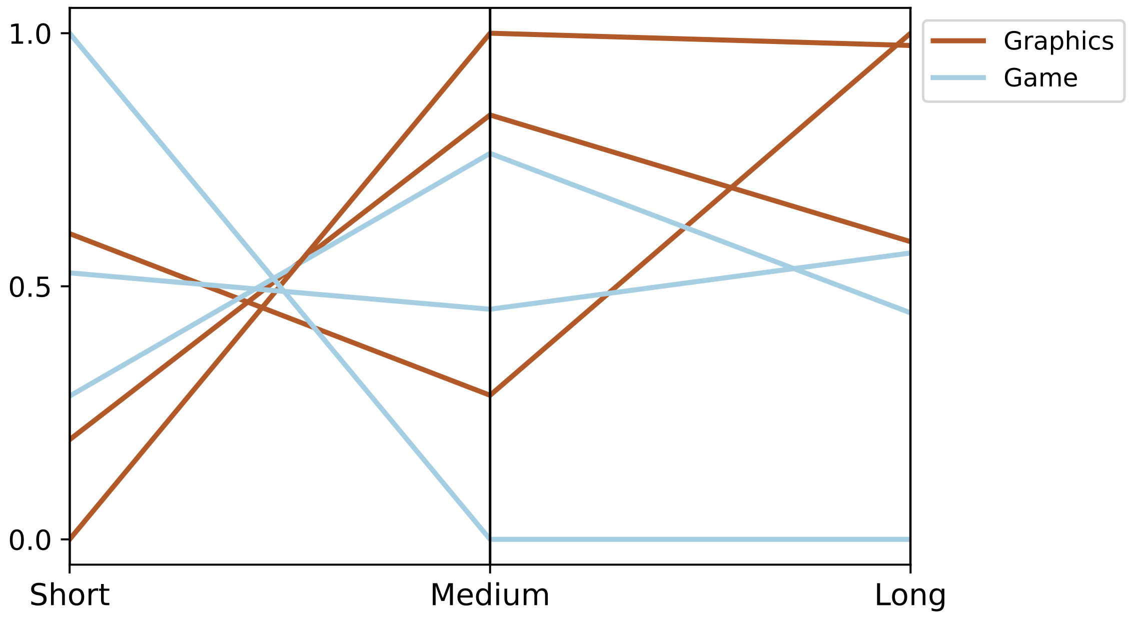


Fig. 4.13: Parallel coordinates of graphics vs. game assignments in the second study.

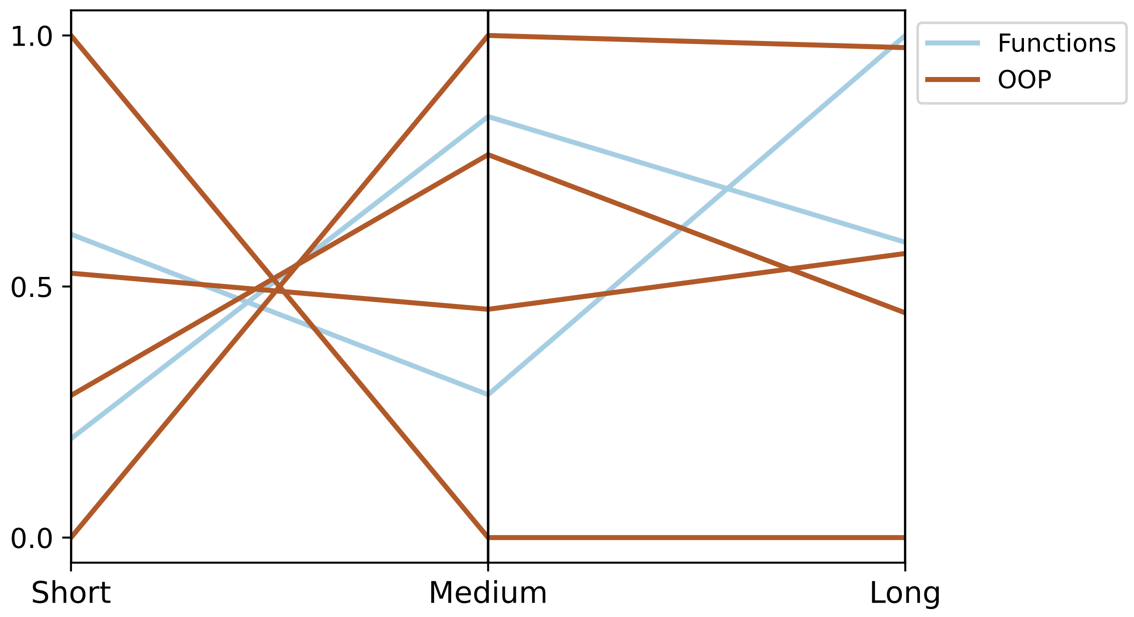


Fig. 4.14: Parallel coordinates of object-oriented programming (OOP) tasks vs. tasks focusing on functions in the second study.

Pause Length	**Load Type**	**R**	**P-value**
Short Pauses	Intrinsic	0.62	0.19
Short Pauses	Germane	0.73	0.1
Short Pauses	Extraneous	0.19	0.72
Medium Pauses	Intrinsic	-0.55	0.26
Medium Pauses	Germane	-0.67	0.14
Medium Pauses	Extraneous	0.21	0.68
Long Pauses	Intrinsic	-0.76	0.08
Long Pauses	Germane	-0.71	0.11
Long Pauses	Extraneous	0.02	0.97

Table 4.6: Pearson R correlation coefficients and P-values for each bin and load type

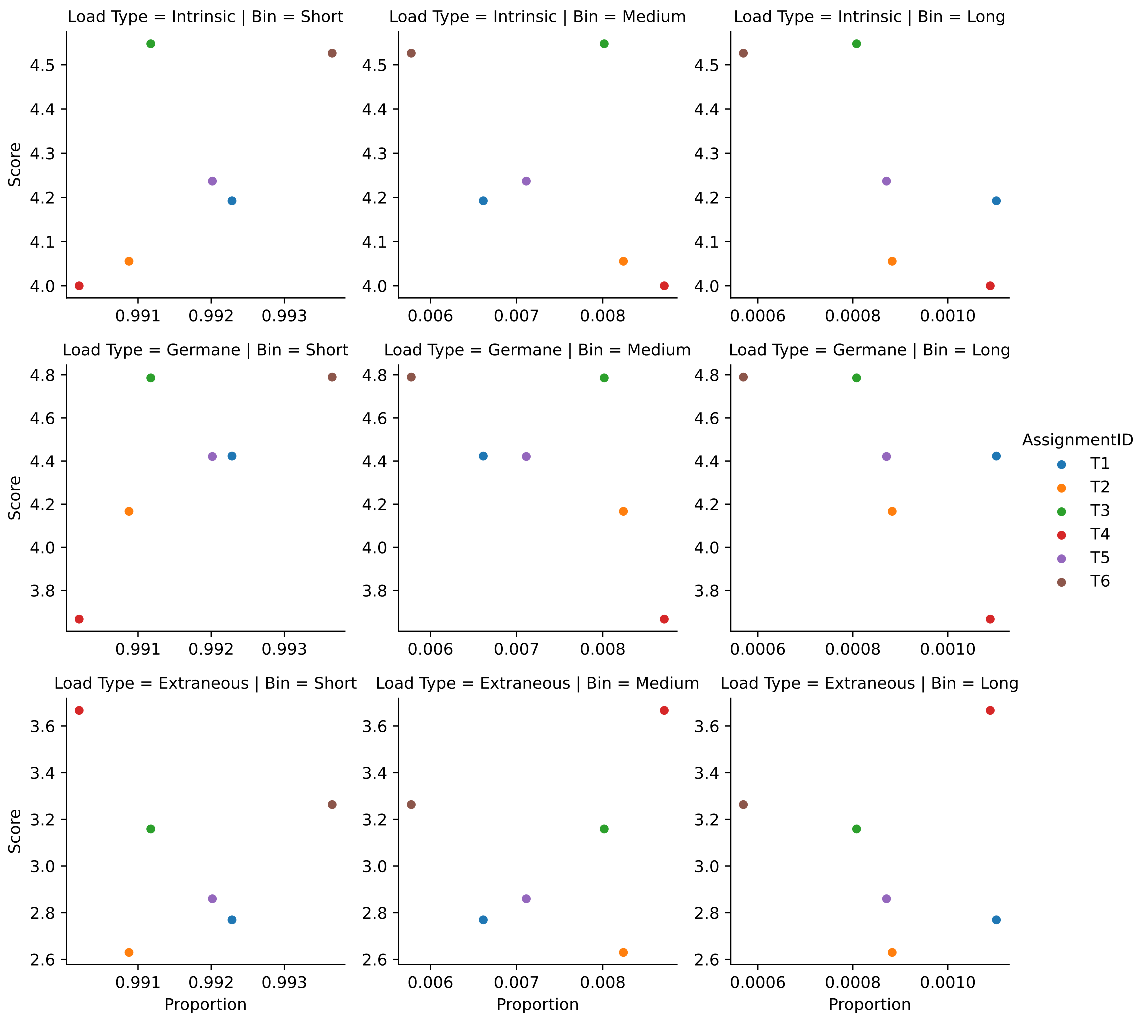


Fig. 4.15: Scatterplot matrix of the median proportion of bin counts (columns) and the scores for each type of cognitive load (rows).

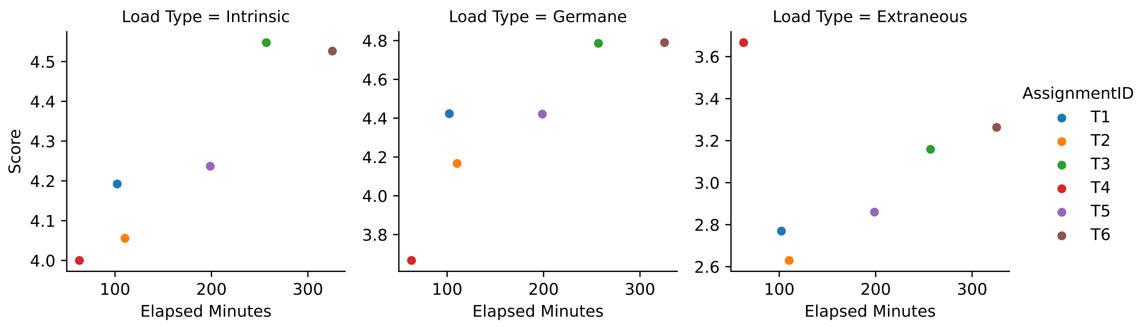


Fig. 4.16: Scatterplot of completion time and load scores.

4.2.3 Completion time and starter code ratio vs. cognitive load

One of the observations in the first study was that assignment completion time may not be related to the cognitive load of an assignment. To test this, the median assignment completion time was also correlated with the cognitive load scores. The scatterplots of completion time and each time of load are given in figure 4.16. Completion time had a significant relation with Intrinsic Cognitive Load (*r = 0.94, p = 0.006*) as well as Germane Cognitive Load (*r = 0.87, p = 0.03*), but not with Extraneous Cognitive Load (*r = 0.08, p = 0.88*). These results suggest that completion time is related to a student's self-reported Cognitive Load.

In addition to the correlation with assignment completion time, the starter code to submitted code ratio was also correlated with the load scores ad-hoc to see what relation an assignment's starter code ratio had with cognitive load. Figure 4.17 shows that a higher starter code ratio lowers Intrinsic and Germane Cognitive Load. Starter code ratio had a significant relation with Germane Cognitive Load (*r = -0.87, p = 0.03*) and a relation that was trending significant with Intrinsic Cognitive Load (*r = -0.78, p = 0.006*), but not with Extraneous Cognitive Load (*r = 0.22, p = 0.68*).

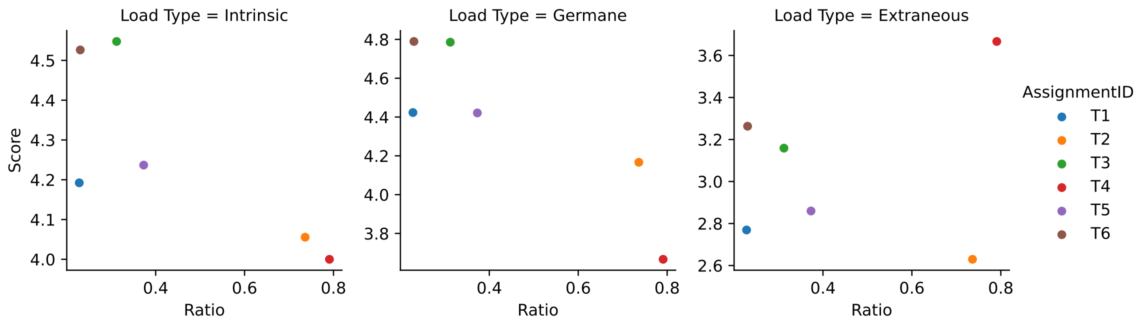


Fig. 4.17: Scatterplot of starter code ratio and load scores.

CHAPTER 5
DISCUSSION

5.1 Assignment pausing behavior differentiation

My 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 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 4.1, Figure 4.4 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 4.1a). 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. Graphics assignments also show similar pausing characteristics in the second study. Figure 4.13 shows a little more variability than study one, but graphics assignments tend to have more medium and long pauses than assignments where students create a game. This suggests that students take more breaks when completing graphics assignments in the second study.

Assignment differentiation is also seen in assignments that present new constructs vs. assignments that do not 4.5. Assignments that present new constructs generally have a low proportion of long pauses. This could signify that students are not experiencing a working memory overload. Assignments that present a new construct tend to provide more scaffolding to help the student learn the construct. The exception is assignment 8, which presents lists. However, we believe this is due to the other characteristics of assignment 8, namely its focus on OOP because a similar assignment (i.e., assignment 7) has similar pausing behavior. Additionally, the other assignments that involve lists do not share a similar pattern with assignment 8. It is also noteworthy that assignment 1, which introduces loops, and assignment 3, which introduces functions, have the second and third highest scaled values of medium pauses. Loops and functions were some of the first programming concepts taught to students. As such, the higher medium pause values for these assignments could be caused by students still learning the syntax of the language, which produces ECL (17; 19). Overall, the parallel coordinate charts suggest that assignments differ in their pausing behavior.

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 4.7 and Table 4.2). 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 OOP 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) in the first study 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 (29). 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 (1). Additionally, the relatively extreme proportion of long pauses signifies that students could be spending more time being disengaged in these assignments (5), which could be caused by a working memory overload and frustration, which stops learning (8).

Assignments 7 and 8 are similar in many ways. Both assignments involve creating a virtual environment where users enter input into 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 OOP 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 OOP, 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 in the first study 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. Interestingly, Assignment 4 (i.e., ”T4”) in the second study is the same as Assignment 6 in the first study, but T4 had the most medium pauses and second most long pauses. This could be because of the change in assignment structure.

Both of the descriptions for the assignments in group 3 (i.e., assignments 7 and 8 in the first study) 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. However, assignments that deal with OOP varied in pausing behavior, as shown in Figure 4.14.

The assignment that caused the most long pauses in the second study is assignment 1 (i.e., ”T1”). This was assignment 4 in the first study. Again, while the same assignment fell in the middle of the medium and long pause bins in the first study, it had the most long pauses in the second. This could be because of the reorganized assignment structure again, or the relatively fewer assignments that were measured in the second study. This assignment asked students to incorporate functions from a separate file into a main file to draw a chessboard. Students figuring out how to incorporate code from other modules/classes may be what causes more medium and long pauses. Further research is needed to solidify this, but these results show that assignments asking students for an intricate implementation taking code from other files result in the highest proportions of long pauses.

5.3 Pausing behavior vs. self-reported cognitive load

The third research question is *Does student pausing behavior correlate with student self-reported cognitive load?* Figure 4.15 shows that IL and GL have a positive linear relation with short pause proportion and a negative linear relation with the medium and long bin proportions. However, these correlations are not statistically significant, but the relation between ICL and long bin proportion is trending significance. ECL does not have a linear relation with any of the bins. These results are different than expected after the first study. The assumption was that the more medium and long pauses an assignment caused, the higher the cognitive load. However, these results show that assignments that cause students to spend fewer breaks have higher self-reported levels of cognitive load.

This is an interesting finding, and further research is needed to validate it. This suggests that breaks, or medium and long pauses, are not due to a cognitive overload. It could just be that the assignment is not engaging enough for the student, or the pauses are just students looking in other sources for assignment help. Or students couldn't simply be more distracted by outside events in these assignments. It is also possible that there could be biases introduced in self-reports. Students were asked to report their perceived cognitive load after assignment completion. One issue with self-reports is that they are usually not in real-time (the questionnaire for this study was not) and students may have different interpretations of the scale (44). It is possible that there could be other methods used to measure cognitive load, such as physiological data (45). However, it is significantly more difficult to get physiological data, and self-report questionnaires are still used widely and accepted by many, particularly with measuring abstract constructs such as cognitive load

(46; 47)

Overall, student self-reported cognitive load does not correlate with the proposed framework using latency analysis, at least what was originally expected. However, this may not invalidate the proposed framework. It could just need further exploration to know what pauses mean. Just tweaking the framework to say that more medium and long pauses correspond to less cognitive load would make the two measurements agree. It could be that assignments already do a good job of keeping students in the ZPD, so the assignments with more shorter pauses have the optimal amount of cognitive load to keep students engaged. Although, since the correlations were not significant in this study, further research is needed to confirm this.

5.4 Assignment completion time vs. self-reported cognitive load

The final research question is *Do assignments that have a longer completion time also have a higher cognitive load?* As mentioned at the start of this thesis, it is a common belief that assignments that take longer are harder. The proposed framework in the first study suggested that this may not always be the case. However, Figure 4.16 shows strong positive linear relations between ICL and assignment completion time, as well as GCL and assignment completion time. The relations are both statistically significant. ECL and assignment completion time do not have a significant relation. However, this is likely due to assignment four in the second study (i.e., T4), which took the least amount of time to complete, but that the highest ECL score. This assignment had the highest ratio of starter code (see Table 4.5). So, it is likely that students needed to spend a significant amount of time understanding the starter code, even though the assignment didn't take that long, relatively. This would explain the higher ECL score, as ECL deals with characteristics that are not directly related to the task, such as assignment design.

Even though T4 had a higher amount of medium and long pauses, it did not have the highest levels of self-reported cognitive load, aside from ECL. This does not support the observation in the first study. However, this could also be because students were judging cognitive load mainly on completion time as they reflected on the assignments. Overall, assignments that have a longer completion time do have higher levels of self-reported cognitive load.

It is also noteworthy that the starter code to finished code ratio of an assignment has a significant negative relation with ICL and GCL (see Figure 4.17). This suggests that the more starter code a student is given, the less cognitive load they will have. This could be because students have to keep less information in their minds during assignments that have a higher starter code ratio than others and it is easier for them to stay in the ZPD with the extra scaffolding. Thus, instructors can provide more starter code if students are struggling with an assignment.

5.5 Threats to validity

There are several threats to the validity of this study. One is the relatively small sample size. The data for each study 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. Additionally, the assignments in the two studies did not exactly match up, so it is not possible to draw direct comparisons. There was also not as much data collected in the second study due to an error in the tool used for data collection. This could also be affecting the analyses and the ability to find statistically significant relations. Next, 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. The results of the self-report questionnaires were also given after assignment completion, as discussed above in Section 5.3, which could provide different results than if they were given in real time. 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, and that students would have different self-reported values with a different combination of assignments, as they would compare different assignments. In the second study, we only collected data on 6 assignments which started about halfway through the course. Missing the beginning assignments when students were very first learning the program could leave out important comparisons in analysis, particularly in the parallel coordinate charts.

CHAPTER 6
CONCLUSION

This thesis has presented two studies. The first presents 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. The second study tests the framework on 6 assignments from another CS1 course, with the addition of a self-report questionnaire to measure cognitive load. This section summarizes the findings from each of our research questions and discusses future work.

RQ1: *Are assignments different from each other in student pausing behavior?* Differences in student pausing behavior can be seen when comparing Turtle graphics and non-Turtle graphics assignments (Figures 4.4 and 4.13) as well as in assignments that present new constructs and those that do not (Figure 4.5). Turtle-graphics assignments share strikingly similar pausing behavior, even though keystroke count and completion time vary between these assignments. With one exception, assignments that present new constructs do not have a high number of long pauses. This could be because these assignments are generally easier to help a student grasp a new concept. Additionally, K-means clustering discovered three groups of assignments (Figure 4.7). A repeated-measures ANOVA showed a statistical difference between a representative assignment from each of the three groups in all of the pause length proportions (i.e., short, medium, and long; Table 4.4). Overall, these results provide evidence that CS1 assignments are different from each other in student pausing behavior.

RQ2: *What types of assignments cause students to take the highest proportion of longer pauses?* The k-means cluster analysis in the first study (see Table 4.2) found a group of two assignments that had a relatively low proportion of short pauses, a higher proportion of medium pauses, and a considerably higher proportion of long pauses. The higher proportion of medium and long pauses indicates that students take more breaks relative to other assignments. The assignments in this group share similar attributes. They both require students to implement detailed implementations of object-oriented programming (OOP) principles, such as classes, methods, private data, etc. These assignments have the highest proportion of long pauses. Long pauses are where students are more likely to be disengaged (5). The assignments that had the highest amount of long pauses in the second study were assignments 1 and 4 (i.e., T1 and T4 4.12). These assignments did not necessarily require students to carry out detailed OOP implementations. Rather, they just had to incorporate functions or classes from a separate file into a main file. So, it could just be the compartmentalization of code that causes long pauses, rather than specific OOP tasks.

RQ3: *Does student pausing behavior correlate with student self-reported cognitive load?* In short, student pausing behavior may correlate with self-reported cognitive load, just not as was expected after the first study. As shown in Figure 4.15, Intrinsic and Germane Cognitive Load (ICL and GCL) had a positive linear relation with pauses 45 seconds in length or less (e.g., short pauses) and a negative linear relation with pauses from 46 seconds to 6 minutes (e.g., medium pauses) and pauses over 6 minutes (e.g., long pauses). This means that the more short pauses a student took, the more likely they were to have higher ICL and GCL scores, while the more medium and long pauses they took, the lower the ICL and GCL scores. However, these relations were not statistically significant, so further research is needed to validate this relation. Additionally, Extraneous Cognitive Load (ECL) had no practical or significant relation with pause length.

RQ4: *Do assignments that have a longer completion time also have a higher cognitive load?* Self-reported levels of ICL and GCL do have a strong, significant linear relation with assignment completion time (See Figure 4.16). There is not a significant relation with ECL. This does not provide evidence that harder assignments do not necessarily take longer. It does confirm that assignments that take longer are perceived by students as having a higher cognitive load.

There is evidence that a relation exists between pausing behavior and cognitive load, though it is not yet known how to explain the relation. Gaining an understanding of the underlying factors will at least benefit instructors in assignment design and could lead to new insights into cognition. The findings also suggest that there may be a difference between perceived cognitive load after the fact, and actual cognitive load, as assignments that took the longest to complete in the second study had the highest ICL and GCL scores.

Future work will include more than six assignments so that a richer comparison can be drawn between assignments. This would allow a deeper dive into the relation between pausing behavior and cognitive load to confirm if more short pauses correspond with higher levels of cognitive load. Additionally, it will be beneficial to examine other real-time measures of cognitive load or give the questionnaire during the assignment so that results do not rely on student recall.

REFERENCES

[1] L. Leppänen, J. Leinonen, and A. Hellas, "Pauses and spacing in learning to program," in *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 2016, pp. 41–50.

[2] J. Edwards, J. Leinonen, and A. Hellas, "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*, 2020, pp. 413–419.

[3] R. Shrestha, J. Leinonen, A. Zavgorodniaia, A. Hellas, and J. Edwards, "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, 2022, pp. 187–198.

[4] J. Urry and J. Edwards, "A framework that explores the cognitive load of cs1 assignments using pausing behavior," in *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 2024.

[5] K. Hart, J. Edwards, and C. Warren, "Accurate estimation of time-on-task while programming," in *In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 708–714.

[6] M. Klepsch, F. Schmitz, and T. Seufert, "Development and validation of two instruments measuring intrinsic, extraneous, and germane cognitive load," *Frontiers in psychology*, vol. 8, p. 1997, 2017.

[7] D. C. Geary and D. B. Berch, "Evolution and children's cognitive and academic development," *Evolutionary perspectives on child development and education*, pp. 217–249, 2016.

[8] J. Sweller, "Cognitive load theory," in *Psychology of learning and motivation.* Elsevier, 2011, vol. 55, pp. 37–76.

[9] J. Sweller, J. J. Van Merrienboer, and F. G. Paas, "Cognitive architecture and instructional design," *Educational psychology review*, vol. 10, pp. 251–296, 1998.

[10] J. Sweller, "Element interactivity and intrinsic, extraneous, and germane cognitive load," *Educational psychology review*, vol. 22, pp. 123–138, 2010.

[11] S. Roussel, D. Joulia, A. Tricot, and J. Sweller, "Learning subject content through a foreign language should not ignore human cognitive architecture: A cognitive load theory approach," *Learning and Instruction*, vol. 52, pp. 69–79, 2017.

[12] N. A. F, "Cognitive load theory in the context of second language academic writing," *Higher Education Pedagogies*, vol. 3, no. 1, pp. 385–402, 2018.

[13] J. Sweller, “Cognitive load theory and educational technology,” *Educational Technology Research and Development*, vol. 68, no. 1, pp. 1–16, 2020.

[14] A. Skulmowski and K. M. Xu, “Understanding cognitive load in digital and online learning: A new perspective on extraneous cognitive load,” *Educational psychology review*, vol. 34, no. 1, pp. 171–196, 2022.

[15] P. Sands, “Addressing cognitive load in the computer science classroom,” *Acm Inroads*, vol. 10, no. 1, pp. 44–51, 2019.

[16] J. H. Berssanette and A. C. de Francisco, “Cognitive load theory in the context of teaching and learning computer programming: A systematic literature review,” *IEEE Transactions on Education*, vol. 65, no. 3, pp. 440–449, 2021.

[17] R. Lister, “Computing education research programming, syntax and cognitive load,” *ACM Inroads*, vol. 2, no. 2, pp. 21–22, 2011.

[18] J. M. Edwards, E. K. Fulton, J. D. Holmes, J. L. Valentin, D. V. Beard, and K. R. Parker, “Separation of syntax and problem solving in introductory computer programming,” in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–5.

[19] J. Edwards, J. Ditton, D. Trninic, H. Swanson, S. Sullivan, and C. Mano, “Syntax exercises in cs1,” in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 2020, pp. 216–226.

[20] D. Weintrop and U. Wilensky, “Comparing block-based and text-based programming in high school computer science classrooms,” *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 1, pp. 1–25, 2017.

[21] L. S. Vygotsky and M. Cole, *Mind in society: Development of higher psychological processes.* Harvard university press, 1978.

[22] R. Wass and C. Golding, “Sharpening a tool for teaching: the zone of proximal development,” *Teaching in Higher Education*, vol. 19, no. 6, pp. 671–684, 2014.

[23] N. Anderson and T. Gegg-Harrison, “Learning computer science in the” comfort zone of proximal development”,” in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 495–500.

[24] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel, “Scaffolding students’ learning using test my code,” in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013, pp. 117–122.

[25] J. Stachel, D. Marghitu, T. B. Brahim, R. Sims, L. Reynolds, and V. Czelusniak, “Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool,” *Journal of Integrated Design and Process Science*, vol. 17, no. 1, pp. 37–54, 2013.

[26] S. O’brien, “Pauses as indicators of cognitive effort in post-editing machine translation output,” *Across languages and cultures*, vol. 7, no. 1, pp. 1–21, 2006.

[27] A. Révész, M. Michel, M. Lee *et al.*, "Investigating ielts academic writing task 2: Relationships between cognitive writing processes, text quality, and working memory," 2017.

[28] J. P. Borst, N. A. Taatgen, and H. van Rijn, "What makes interruptions disruptive? a process-model account of the effects of the problem state bottleneck on task interruption and resumption," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 2971–2980.

[29] J. Y. Lee, J. Donkers, H. Jarodzka, G. Sellenraad, and J. J. Van Merriënboer, "Different effects of pausing on cognitive load in a medical simulation game," *Computers in Human behavior*, vol. 110, p. 106385, 2020.

[30] A. Vihavainen, M. Luukkainen, and P. Ihantola, "Analysis of source code snapshot granularity levels," in *Proceedings of the 15th annual conference on information technology education*, 2014, pp. 21–26.

[31] J. Leinonen, K. Longi, A. Klami, and A. Vihavainen, "Automatic inference of programming performance and experience from typing patterns," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 132–137.

[32] J. Edwards, K. Hart, and C. Warren, "A practical model of student engagement while programming," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 2022, pp. 558–564.

[33] J. Leinonen, F. E. V. Castro, and A. Hellas, "Time-on-task metrics for predicting performance," *ACM Inroads*, vol. 13, no. 2, pp. 42–49, 2022.

[34] A. Carbone, J. Hurst, I. Mitchell, and D. Gunstone, "Characteristics of programming exercises that lead to poor learning tendencies: Part ii," *ACM SIGCSE Bulletin*, vol. 33, no. 3, pp. 93–96, 2001.

[35] T. J. Feldman and J. D. Zelenski, "The quest for excellence in designing cs1/cs2 assignments," *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 319–323, 1996.

[36] J. Wolfe, "Why the rhetoric of cs programming assignments matters," *Computer Science Education*, vol. 14, no. 2, pp. 147–163, 2004.

[37] L. Layman, L. Williams, and K. Slaten, "Note to self: make assignments meaningful," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, 2007, pp. 459–463.

[38] D. E. Stevenson and P. J. Wagner, "Developing real-world programming assignments for cs1," *ACM SIGCSE Bulletin*, vol. 38, no. 3, pp. 158–162, 2006.

[39] R. Garcia, "Assignment presentation framework for cs1 programming problems," in *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2021, pp. 1–9.

[40] C. L. Kussmaul, "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*, 2008, pp. 873–876.

[41] P. Kinnunen and B. Simon, "My program is ok–am i? computing freshmen's experiences of doing programming assignments," *Computer Science Education*, vol. 22, no. 1, pp. 1–28, 2012.

[42] J. Edwards, K. Hart, R. Shrestha *et al.*, "Review of csedm data and introduction of two public cs1 keystroke datasets," *Journal of Educational Data Mining*, vol. 15, no. 1, pp. 1–31, 2023.

[43] J. Edwards, "2021 CS1 Keystroke Data," 2022. [Online]. Available: https://doi.org/10.7910/DVN/BVOF7S

[44] G. Matthews, J. De Winter, and P. A. Hancock, "What do subjective workload scales really measure? operational and representational solutions to divergence of workload measures," *Theoretical issues in ergonomics science*, vol. 21, no. 4, pp. 369–396, 2020.

[45] P. Vanneste, A. Raes, J. Morton, K. Bombeke, B. B. Van Acker, C. Larmuseau, F. Depaepe, and W. Van den Noortgate, "Towards measuring cognitive load through multimodal physiological data," *Cognition, Technology & Work*, vol. 23, pp. 567–585, 2021.

[46] R. Pekrun, "Self-report is indispensable to assess students' learning. frontline learning research, 8 (3), 185–193," 2020.

[47] L. K. Fryer and D. L. Dinsmore, "The promise and pitfalls of self-report: Development, research design and analysis issues, and multiple methods," *Frontline Learning Research*, 2020.

APPENDICES

APPENDIX A

Cognitive Load Questionnaire

The following questionnaire was given to students after each assignment in the second study to measure the intrinsic, extraneous, and germane cognitive load. This measure was adapted from the measure created and validated by Klepsch et al.(6). The questions are as follows:

1. *This assignment required me to keep many things in mind simultaneously.*
2. *This assignment was very complex.*
3. *This assignment required me to stay highly engaged.*
4. *This assignment required me to think intensively about what things meant.*
5. *It was exhausting to find the pertinent information for this assignment.*
6. *The design of this assignment was inconvenient for learning.*
7. *It was difficult to recognize and link crucial information for this assignment.*

The questions measure different types of cognitive load. Questions 1-2 measure ICL, questions 3-4 measure GCL, and questions 5-7 measure ECL. Each question was answered on a Likert scale of 1 (Strongly Disagree) to 5 (Strongly Agree).

1 - *Strongly Disagree*

2 - *Disagree*

3 - *Neutral*

4 - *Agree*

5 - *Strongly Agree*

CURRICULUM VITAE

Joshua O. Urry

Published Conference Papers

- A framework that explores the cognitive load of CS1 assignments using pausing behavior, Joshua Urry and John Edwards, in *55th ACM Technical Symposium on Computer Science Education (SIGSCE)*, 2024.