THESIS

STUDENT RESPONSE TO TEACHING OF MEMORY CUES AND RESUMPTION STRATEGIES IN COMPUTER SCIENCE CLASSES

Submitted by

Noah John

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2015

ABSTRACT

STUDENT RESPONSE TO TEACHING OF MEMORY CUES AND RESUMPTION STRATEGIES IN COMPUTER SCIENCE CLASSES

Programming is a creative process that requires the ability to concentrate and juggle multiple concepts simultaneously in one's mind. Existing research shows there is a tangible cost when a programmer is interrupted as the programmer must recover the context of his work and refocus on the task at hand. However, computer science (CS) students are rarely taught about interruptions and how to manage them. Instead, teaching tends to focus only on technical concepts. In addition, there is little research on interruptions with respect to CS students. Therefore, our research examines what happens when CS students are taught about interruptions and how to cope with them.

The objective of this research is to determine if CS students are affected by interruptions, what knowledge CS students possess regarding memory cues and resumption strategies, and what their opinion is of this material. To answer these questions, we began by conducting a pilot study with fifteen students that consisted of an initial questionnaire, a seminar on memory cues and resumption strategies, and a follow-up questionnaire. After positive findings, we conducted a user study with approximately two-hundred undergraduate CS students. Our user study was comprised of a modified initial questionnaire, an identical seminar on memory cues and resumption strategies, and a modified follow-up questionnaire. Our results demonstrate that CS students are affected by interruptions, but 73% of students report not knowing methods to mitigate them. After learning about memory cues and resumption strategies, students report that the material is useful and that they want to study it. Their most significant feedback is that they have a strong desire to include these techniques in CS curriculums, reporting a mean score of 7.78 out of 10, where 0 signifies strong disagreement and 10 signifies strong agreement.

ACKNOWLEDGEMENTS

I would like to thank the participants of the study for their time, the generous Colorado State University faculty for opening the door to their classrooms, Dr. Jamie Ruiz for his guidance, and my colleagues for providing thoughtful feedback. I want to thank Dr. Wim Böhm and Dr. Ed DeLosh for taking the time and making the effort to be on my committee. In addition, I want to thank my great friends David Pilo, Jacob Skubal, and Rusty Dyer for making me laugh and keeping my ego in check. Finally, I must thank my parents – you guys are the best.

This thesis is typset in LaTeX using a document class designed by Leif Anderson.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

Chapter 1. Introduction 1

Chapter 2. Background 4

2.1. Interuption cost and effects 4

2.2. Reasons for interruptions 5

2.3. Interruption recovery 6

2.4. Coordinating interruptions 7

2.5. Specialized interruption research 8

Chapter 3. Pilot Study 9

3.1. The initial questionnaire 9

3.2. The seminar on memory cues and resumption strategies 10

3.3. The follow-up questionnaire 15

3.4. Results 16

3.5. Discussion 24

Chapter 4. User Study 27

4.1. The initial questionnaire 28

4.2. The seminar on memory cues and resumption strategies 29

4.3. The follow-up questionnaire 29

4.4. Results 29

4.5. Discussion 35

Chapter 5. Limitations 38

Chapter 6. Future work 39

Chapter 7. Conclusion 40

Bibliography 42

LIST OF TABLES

3.1 Pilot study: Summary of answers to part one of the initial questionnaire. 16

3.2 Pilot study: Summary of answers to part one of the follow up questionnaire. 22

3.3 Pilot study: Summary of answers to part two of the follow up questionnaire. 22

4.1 User study: Top contributors to productive/unproductive work for CS students. . . 30

4.2 User study: Summary of answers to part one of the follow-up questionnaire. 32

4.3 User study: Summary of answers to part two of the follow-up questionnaire. 33

4.4 User study: Results of Analysis of Variance tests on the quantitative self-reported responses from the follow-up questionnaire. 35

LIST OF FIGURES

3.1 Example of continuous rating scale question from follow-up questionnaire. 16

4.1 User study: Student responses to the question *"Do you do anything prior to stepping away from an assignment that helps you refocus when you return?"* 31

4.2 User study: Blox plot of follow-up answers 32

CHAPTER 1

Introduction

Computer science (CS) students are taught a myriad of topics over the course of their education. In an undergraduate curriculum, it is common to see classes on programming languages, programming paradigms, data structures, algorithms analysis, and operating systems. As a student, it is important to learn the fundamental concepts taught in these classes to become a knowledgeable software engineer. Furthermore, a large portion of the work CS students do is programming-related. Whether the language is Java, C++, Scheme, or assembly, programming represents a majority of a CS student's homework and often times is the primary benchmark for a student's performance. And while the technical details and concepts, such as object hierarchy or C++ pointer syntax, are conveyed to students in class, significantly less is conveyed about the art of programming. At its core, programming requires strong logical reasoning, the knack for solving problems, and smart design skills. However, it is also a creative process that requires the ability to concentrate and juggle multiple concepts simultaneously. Additionally, there is a tangible cost when a programmer is interrupted as the programmer must recover the context of his work and refocus on the task at hand. On average, industry software engineers need fifteen minutes to recover from an interruption and return to a focused and productive state [1]. While there has been extensive research on the topic of interruptions, their effects, and methods for minimizing their impact [2], very little has reached CS curriculums.

In contrast, consider an English curriculum. One of its goals is to produce proficient and eloquent writers capable of composing various types of prose. Writing is recognized as a creative process that requires concentration and the managing of multiple ideas at the same time. English students, along with a majority of the general public, are familiar with the term "writer's block", which refers to a condition when an author is unable to produce new material. Furthermore, writing teachers dedicate some of their time to discussing this barrier and how to overcome it. Informal discussion of techniques and experiences has been deemed useful by the writing community and writer's block is one of the concepts covered in an English curriculum. While writer's block is not analogous to a programmer being interrupted and then having to resume a task, there are parallels. In both cases, the person is unproductive for a period of time and must find a way to overcome this state. Additionally, as a writer must remember his train of thought, a programmer must recover his context and his intended next step. We discuss these topics with English students. But how would CS students react if they were taught about interruptions and how to cope with them?

The objective of this thesis is to determine if CS students are affected by interruptions, what knowledge students possess regarding memory cues and resumption strategies, and what their opinion is of learning this material. We began by conducting a pilot study of 15 undergraduate students. First, we surveyed their programming habits, their work, and their previous knowledge on memory cues and resumption strategies. Next, we gave a seminar on interruptions, memory cues, and resumption strategies to these same students. Finally, after their subsequent programming assignment, we conducted a follow-up survey to gather their opinions on the seminar and examine if it had any perceived effect on their work. Encouraged by a positive self-reported effect of the seminar in the pilot study, we conducted a more formal user study with approximately two-hundred undergraduate CS students of varying ages, class standing, and programming experience over the course of half a semester.

The user study results demonstrate that CS students are affected by interruptions and that they report the same resumption lag as industry engineers. We also see that 73% of

students report not knowing about memory cues and resumption strategies. Finally, we find that students believe the material is useful and have a strong desire to learn these techniques.

CHAPTER 2

Background

There has been extensive research examining the effects of interruptions on productivity. A critical and foundational paper was published by McFarlane and Latorella [3] which thoroughly reviewed existing literature to argue its fundamental place in HCI research and how vitally important it is to understand interruptions and their effects as technology continues to evolve. They began by analyzing three application domains to show the necessity for considering interruptions in informed design decisions. They continued by summarizing their respective research that formalized and categorized interruptions with respect to tasks and summarizing their respective research into managing interruptions. They ultimately concluded with a discussion of existing strategies for mitigating the cost of interruptions and proposed that UI design was the most promising strategy, hence the importance of interruption research in HCI. Generally, research falls into four areas: interruption cost and effects, reasons for interruptions, interruption recovery, and coordinating interruptions. These four areas are summarized below.

2.1. Interuption cost and effects

One area of research has centered on the cost and effects of interrupting a person performing a task. Adamczyk and Bailey [4] conducted a user study to determine which are the least costly moments to interrupt someone's task. They found the best moments to interrupt a person were when the person was switching between well-understood and defined tasks. Hodgetts and Jones [5] [6] studied the effect of interruptions during different points in a problem-solving process. Their results showed that brief interruptions still disrupted workers and longer interruptions produced a stronger negative effect on workers. In addition, they found the period between being alerted to an interruption and reacting to an interruption was critical to workers as it could be used to encode cues that aid task resumption later. They then leveraged these findings to inform the design of an attention manager system [4]. Robertson *et al.* [7] studied the effects of *immediate-style* interruptions and *negotiated-style* interruptions on users debugging spreadsheet errors. Immediate-style interruptions are interruptions that require user action. Conversely, negotiated-style interruptions were defined in McFarlanes classification of interruptions [8] as interruptions that inform the user of a pending message but do not force them to acknowledge it immediately. Comparing the two styles, Robertson *et al.* found that negotiated-style were more effective.

Iqbal and Bailey [9] identified concepts from cognitive psychology that could be used to estimate the Cost of Interruption (COI) and Salvucci *et al.* [10] proposed a theoretical framework for defining the states of a multitasking continuum which included the states of interruption and resumption. Marulanda-Carter and Jackson [11] studied e-mail usage in a large international car rental company, measuring how e-mail interruptions affected productivity and the prevalence of e-mail addiction. They found between 12-15% of workers were e-mail addicts and that e-mail interruptions caused tasks in their study to require a third more time to complete.

2.2. Reasons for interruptions

Some researchers have studied the reason for interruptions by observing the behaviors of people in an office and noting interruptions and their motivations. Czerwinski *et al.* [12] conducted a diary study of eleven experienced office software users and found that "returned-to" tasks were most difficult to switch to, required the most time to complete, and experienced the most interruptions (due to the task length). Szstek and Markopoulos [1] observed the behaviors of people in an office and noted interruptions and their motivations. The study focused on face-to-face communication and highlighted factors such as urgency, employee hierarchy, and perceived availability.

Harr and Kaptelinin [13] thoroughly summarized existing research of interruptions and underlined the lack of attention paid to the social component of interruptions and the various dynamics that are in play. The paper identified four variables that affected the likelihood of an interruption occurring and proposed four related "ripple effects" that add additional cost to an individual interruption. In follow-up work, Harr and Kaptelinin [14] verified the effect of social context in a person's decision to interrupt someone else.

Jett and George [15] defined four types of interruption: intrusion, break, distraction, and discrepancy and detailed the positive and negative consequences for the interrupted person. Jin and Dabbish [16] defined a typology for self-interruptions. They ultimately identified seven types of self-interruption: adjustment, break, inquiry, recollection, routine, trigger, and wait. For each type, they observed their effects and duration.

2.3. Interruption recovery

Another area of research has been how one recovers from an interruption. Altmann and Trafton [17] studied the recovery process of users that were interrupted during the course of playing a computer strategy game and Jackson *et al.* [18] looked at the recovery time for e-mail interruptions of users of Microsoft Outlook in a software company.

There have been GUI prototypes that attempted to intelligently alert and interrupt the user (where the software itself has to interrupt the user) while assisting the user in quickly resuming his previous task after dealing with the interruption [19] and a prototype program that helps users organize related items on the Windows taskbar [12].

Parnin and DeLine [20] conducted a survey of software engineers to compile a list of cues that programmers use to resume a task that had been interrupted and then tested the usefulness of three memory cues: note-taking, a content timeline, and a degree-of-interest (DOI) treeview. Results showed programmers favored the content timeline overall, but found the DOI treeview useful for source code tasks. In follow-up research, Parnin and Rugaber [21] looked at the techniques and strategies programmers use to resume an interrupted task. They found only 10% of programmers resumed coding in less than and minute and the remaining 90% typically needed 5-45 minutes to resume. Resuming involved performing non-coding activities to help regain task context before coding again.

2.4. Coordinating interruptions

The notion of coordinating interruptions is rooted to research done by Latorella [22] and McFarlane [2] where "coordinating interruption" was defined as the method one takes in interrupting someone's task. McFarlane [2] empirically evaluated four methods (immediate, negotiated, mediated, and scheduled) for coordinating interruptions and its results suggested that a method's effectiveness depended on the nature of the task being interrupted. He then built on this paper to evaluate four UI design solutions for coordinating interruptions [8].

Gievska *et al.* [23] looked at computer-mediated coordination and conducted an experiment that compared the performance of users that were interrupted randomly to that of users that were interrupted strategically using an interruption mediator. This paper established that workers were less distracted, were less annoyed, and found their work more manageable with mediated interruptions. Also, Kern *et al.* [24] had their study participants rank the personal and social disturbance level of various interruptions in differing environments. He then experimented with sensors to evaluate a person's personal and social interruptability and achieved a 98.1% recognition score for social interruptability. Palanque *et al.* [25] analyzed interruptions using model-based analytical techniques that could be used in designing interruption-tolerant systems. To demonstrate, the authors used a case study to compare two interaction techniques, *Speak and Drop* and *Drag and Drop*, and found the latter was more resistant to interruptions.

2.5. Specialized interruption research

In parallel to the topics already mentioned, in should be noted that a significant portion of research into interruptions focuses on two disciplines: software engineering [26] [20] [21] [7] [27] and e-mail usage [18] [11].

We were unable to find research that investigated the resumption lag of CS students or the effect of teaching them about interruptions. This formed the basis of our motivation to pursue this study.

CHAPTER 3

Pilot Study

Our pilot study had three components: an initial survey, a seminar on memory cues and resumption strategies, and a follow-up survey. The study involved 15 undergraduate computer science students (12 male, 3 female) attending Colorado State University (CSU). All students were enrolled in the course, CS253: Problem Solving with C++, which is the final course of the five-course programming sequence that all students begin with in the computer science curriculum in the CSU CS department. In addition to teaching students C++, the course serves to complete the process of training a professional programmer, which is the objective of the five-course programming sequence. Students in the class were given the opportunity to volunteer in the study and take a seminar, but they were not told what the seminar's topic was, only that it was related to computer science. Those that elected to participate were not compensated.

3.1. The initial questionnaire

The first questionnaire was given at the beginning of the seminar and consisted of 13 questions. Students completed the survey in 5 to 10 minutes.

The first four questions focused on productive and unproductive sittings during a programming assignment. A 'sitting' was defined as the period of time a student works on an assignment at one time without taking breaks (i.e. a student 'sits down' to do work, and 'gets up' to take a break). Students were asked about sitting length and what caused a sitting to be productive or unproductive.

The next three questions dealt with the concept of being 'in the zone', which is an informal description for being in a highly focused and productive state. Students were asked how long it took them to get in the zone when starting and resuming an assignment and how long they could typically stay in this state.

Questions 8, 9, and 10 asked students about their programming habits and focused on what they did to concentrate while programming. They were asked how they went about beginning and resuming a programming assignment and if they did anything to maintain focus while programming.

Finally, as a nod to the pending seminar, students were asked specifically whether they did anything prior to stepping away from programming that helped them refocus faster when they resumed working and if they had heard or read about memory cues and resumption strategies.

Once the questionnaires were competed, the seminar started.

3.2. The seminar on memory cues and resumption strategies

Each participant attended one seminar. In order to accommodate the fifteen participants' schedules, there were five seminars held by the same instructor. There were one to four participants in each seminar. The format was informal, students were encouraged to interrupt with questions, and a white board was heavily used by the instructor. All seminars covered the same material, and lasted 30 to 40 minutes depending on the number of student questions.

3.2.1. Effect of Interruptions. The seminar began with the instructor asking students about their knowledge of memory cues and resumption strategies, which was immediately followed by the instructor introducing the fact that there had been extensive research into the effect of interruptions and their cost to programmers. The major fact highlighted was that research had shown, on average, a programmer requires 15 minutes to refocus after being interrupted.

3.2.2. Memory Cues and Resumption Strategies. The instructor continued by describing various types of interruptions: in-person questions, phone calls, text messages, emails, social media alerts, and self-imposed breaks (like going to the bathroom or getting food). Then, he introduced the fact that, in addition to the research on the effect of interruptions, there had also been research that had asked programmers how they minimize the effect of interruptions. Students were told that memory cues and resumptions strategies were ways programmers refocus faster. While they were shown to be helpful, there was no one technique that was better than another. Choosing a useful technique depended on the programmer's personal preferences. Hence, according to the instructor, the goal of seminar was to share some of those techniques with students.

3.2.3. Brain Dump. Before going into specific memory cues and resumption strategies, the instructor first discussed the high-level idea behind them, which he called a 'brain dump'.

"At its core," he said,"the idea is to dump the contents of your brain before you stop working in such a way that you can easily digest it and reload it into your brain when you resume. This is analogous with what an operating system does when it switches processes; it saves off (or dumps) the values of all the memory registers of the current process so its state is saved for when it runs again. While this example is very mechanical, conceptually it parallels this notion of a brain dump."

3.2.4. Resist Coordinating Interruptions Immediately. The instructor also stressed that, whenever possible, students should resist the temptation to address an interruption immediately and take a short period of time (even a few seconds would suffice) to do something that could help them refocus faster when they resumed. As an example, he mentioned that when asked a question while working, students could reply "give me a few moments" and use the time to mark their work and progress, as this could be helpful to them, yet also acceptable in the social context.

3.2.5. Note Taking. The first technique presented was note taking. “Notes can assume different forms,” the instructor said. “They can be a list on a piece of paper or a collection of diagrams. It completely depends on the preference and style of the programmer.”

He continued with a personal example, “When I worked in industry, I had a whiteboard at my desk. I would maintain an ordered list of tasks and reminders to help keep my focus. I would go so far as to give commands to my ‘future-self’. For example, I would write something like ‘after lunch, look at the file player.cpp and add speed logic on line 198 in the function MovePlayer()’. Many times, when following one of my directions, I would start by wondering why I wrote what I did, only to realize and remember thirty seconds later.”

3.2.6. Code Commenting. The next technique covered was commenting code. While at first glance this seemed identical to note taking, the instructor pointed out that code commenting is sometimes preferred by programmers because it gives them the ability to annotate within the context of the code itself. They can insert TODOs without having to describe where in code to work. They can use pseudo-code to quickly outline an algorithm while it is fresh in their mind. They can scaffold entire blocks of code before actually implementing them, and use the scaffold to direct their work. The instructor also mentioned that commenting code has the advantage of being something quick and dirty for marking progress, but can be less effective if the memory cues need to cover multiple source files and/or if the editor has to be closed. A programmer might not remember what he or she commented before leaving, and hence not benefit from the comments when resuming.

3.2.7. Window Arrangement. Continuing that thread, the next technique was spatial window arrangement. The technique involved positioning all relevant windows (code editors, compiler output, internet reference material, etc.) in a way that highlighted the task at hand and the current state of the programmer's work. The instructor commented that this was a risky strategy because it assumed the computer's state would not change and because there were many ways the position of the windows could be disrupted without the programmer's control or knowledge. However, depending on the format of the information in each window and the ability of the programmer to associate position with the information's priority, it could serve as a good visual memory cue for remembering the previous work state.

3.2.8. Roadblock Cues. Roadblock cues were the next technique introduced in the seminar. This term was used by Parnin [21], and usually refers to when a programmer intentionally inserts a compile error into his code before turning his attention away from his work. Then, upon returning to work, the programmer compiles out of habit to help regain his context. In addition to presenting students this example, the instructor also pointed out how this technique does not rely on the programmer's workspace to be static during a break. The goal of mentioning this detail was to impart the differences among various memory cues and resumption strategies so students could better evaluate which ones could suit their programming style.

3.2.9. Selective Suspension. The final technique focused on selecting a good moment to stop working. Borrowing from a technique Ernest Hemingway employed as a writer [28], students were told that some find it advantageous to avoid stopping at the end of something (e.g. in the case of writers, after completing a chapter) and instead of stopping in the middle of something. Meaning, if a student finished a task, it followed that he should begin the next task before ultimately stepping away. The logic behind this technique is that it is easier for a person in the middle of a task to remember what he was doing and harder for a person that has finished a task to remember what he did and determine what the next step should be. To relate to programming work, the instructor likened a chapter in a book to an encapsulated chunk of code in a program.

3.2.10. Student Questions. Given the informal nature of the seminar, students asked questions and other related topics were covered. One topic that was discussed in all the seminars was using white noise to help maintain focus while programming. Another topic that came up was the use of rituals to help prime the brain. By establishing a habit when performing a task, one could associate this ritual with a given task and thus facilitate a certain mindset by performing the ritual. The instructor did not affirm or deny the validity of the related topics that arose during discussions in the seminar.

3.2.11. Concept Maps. To conclude the seminar, the instructor introduced participants to concept maps and their relation to the brain. To start, he wrote the course name 'CS253' on a whiteboard and asked students to give him concepts that they thought of when thinking of their CS253 course. After a few answers, the whiteboard contained the following nodes: 'CS253', 'C++', 'classes', 'objects', 'pointers', and 'const'.

The instructor continued with the example, "So the idea is that we can draw lines between related concepts. For example, when I think of pointers, I also think of objects because pointers can point to objects. Hence, I draw a line between the two. Also, I know we can have const pointers so I'll draw a line between those two as well. As you can imagine, concept maps can grow large quickly and have many connections. However, what this suggests is that people learn, at least in part, by associating new concepts with concepts that have already been learned."

The instructor then wrote 'Java' on the whiteboard, "In reality, when you started learning C++, you already had learned a lot of similar concepts already when you studied the Java programming language. You didn't have to build this from scratch."

3.2.12. Final Words. Finally, the instructor summarized what they had covered and mentioned that the core idea was that by knowing how the brain works, one can leverage that knowledge to help oneself when performing cognitive tasks, namely refocusing and concentrating. Before leaving, students were told that they would receive a follow-up questionnaire after they had completed their next programming assignment.

3.3. The follow-up questionnaire

The second survey was given a little over a week after the seminar and after students had completed a programming assignment. The survey consisted of two parts. Part one was comprised of six statements and six continuous rating scale questions, each having 'strongly disagree' on the left and 'strongly agree' to the right. Participants were asked to mark a vertical line through the horizontal line scale to indicate their opinion for each of the six statements. We codified their answers by measurement, which gave us real numbers on the continuous interval $[0, 10]$. Figure 3.1 shows one of the questions. The statements centered on the seminar's effect on students. More details can be found in the subsequent Results section. The second part of the questionnaire had six questions. The first four asked for details about the effect of the seminar. The fifth question asked students if they wanted to see a summary of the study's results. The last question was a spot for participants to leave additional comments.

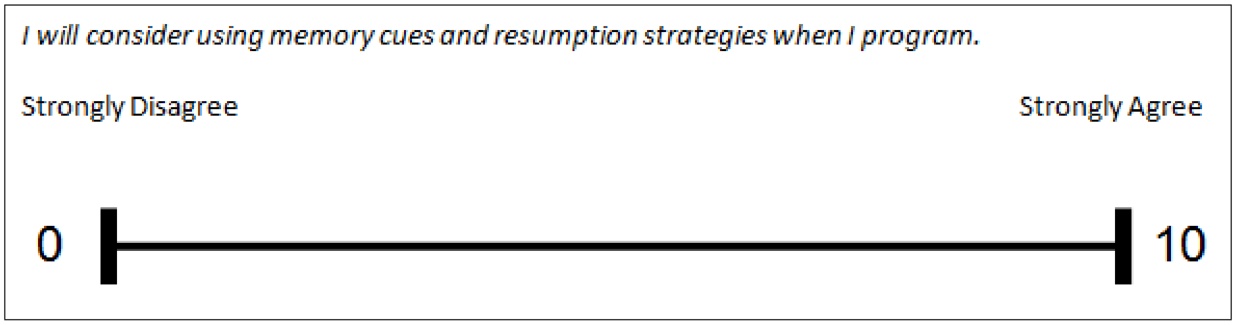
I will consider using memory cues and resumption strategies when I program.
Strongly Disagree
Strongly Agree
0
10

FIGURE 3.1. Example of continuous rating scale question from follow-up questionnaire.

3.4. RESULTS

This section discusses the results of the two questionnaires, identifies themes among students, and analyzes the seminar's effect on students.

3.4.1. INITIAL QUESTIONNAIRE.

The questionnaire had two parts. The first part asked students about the time cost in various scenarios, while the second part delved into habits and behavior. Below, in Table 3.1, is a summary of the results to the first part.

TABLE 3.1. Pilot study: Summary of answers to part one of the initial questionnaire.

Question	**Mean**	**SD**
Number of sittings needed finish a programming assignment	4.07	2.31
Length of a productive sitting	2.98hrs	1.18hrs
Length of an unproductive sitting	1.09hrs	1.23hrs
Time needed to get in the zone when starting a programming assignment	25.9min	20min
Time needed to get in the zone when resuming a programming assignment	12.3min	8.89min
Time to stay in the zone	2.87hrs	1.30hrs

Note that responses to the second (productive sitting) and last (staying in the zone) questions are consistent, which we felt supported the internal validity of the survey. Based on the results of the second part of the questionnaire, we were able to identify themes amongst the study's participants.

3.4.1.1. *Little prior knowledge of memory cues and resumption strategies.* Eleven of the fifteen participants had not heard or read about memory cues or resumption strategies. Three students had heard of these terms, but only in passing. As one of them wrote,

P5: *"[I've heard or read about] some, but never looked too deep or used any."*

The remaining student had read about memory cues via study habit articles and management classes. He provided an example of something he had learned,

P10: *"Having a 'work' or 'coding' playlist gets your mind into the field of that playlists [sic] type."*

3.4.1.2. *Intuitive use of memory cues and resumption strategies.* Even though a majority of students had not heard of memory cues or resumption strategies, all but three of them described using these techniques when they were planning to stop coding. When asked if they did anything prior to stepping away from an assignment that helped them refocus faster when they returned to work, the most common response (11 out of 15) was that they left comments for themselves. The content of the comments varied depended on the programmer, but all of them were clearly used to help regain context more quickly.

P15: *"Generally commentation [sic]. Just reminders or concerns, or specific tests to keep in mind."*

P14: *"[I] make comments in my code explaining where I am and what I'm doing."*

P7: *"I usually put comments in the code to either note where I stopped and what problems might be unresolved."*

P10: *"I place thought process comments and say what I want to be done."*

These answers are consistent with what Parnin [21] found were the suspension strategies of programmers in his evaluation of memory cues. Using comments to help regain context is a common technique employed by programmers.

Additionally, three students remarked that they tried to take their breaks after finishing a task.

> **P1:** *"[...][I] generally only try to quit when a method is done."*
>
> **P5:** *"If I'm in the middle of something, I leave detailed notes, TODOs, etc. Otherwise, I try to only stop after completing some chunk to be used later."*

This is an interesting response because it conflicts with one of the suspension strategies (the Hemingway example) discussed in the seminar. Note this questionnaire occurred at the start of seminar, so students had not been presented any techniques yet.

3.4.1.3. *Maintaining focus.* Thirteen of the fifteen participants stated that they did something to help maintain focus while working on a programming assignment. The most common answer (8 of 15) was that they listened to music to help block out distractions. Four students cited food and beverage as an aid to maintaining focus. Four other students mentioned they used breaks from programming to maintain focus. While some used breaks to relax the mind, breaks did not have to be passive.

> **P9:** *"[I take] short breaks to clear my head or think about the problem without making hasty changes."*
>
> **P12:** *"[I] pace, think aloud, write things down on paper."*

3.4.1.4. *Factors contributing to productive/unproductive work.* The two most mentioned factors that affected productivity were distractions (6 of 15) and understanding (6 of 15). Distractions were primarily related to environment, with noise, other people, non-work technology, and mindset being the main causes for distractions.

P1: *"productive - being at home with little distraction [...] unproductive - being in the lab when friends are there [...]"*

P6: *"Internet/email/phone causes unproductive [sic]."*

P8: *"[Productivity depends on] how distracted I get (i.e. going on Reddit [a social news and entertainment website])."*

P5: *"Either a lack of focus/distractions, or an unclear plan tends to ruin things."*

We were encouraged by the comments on distractions because in many of the examples cited, the distraction was synonymous with a type of interruption and the seminar would directly deal with this topic. This provides some evidence that computer science students are affected by interruptions while doing their assignments. While to some this might sound trivially obvious, we felt it was important to have some supporting feedback from students. As mentioned in the last quote, understanding the assignment and having a plan also contributed to the productivity of the participants. Other factors mentioned were fatigue (2 of 15) and buggy code (2 of 15).

3.4.1.5. *How students start.* Students were asked what they did at the beginning of a programming assignment. Just over half of the participants (8 of 15) said that they used pencil and paper to start working. They outlined ideas or constructed a plan in writing to help formulate a solution to their problem.

P1: *"[I] plan out what the program does, work out any tricky methods on paper, [and] outline the program."*

P3: *"[I] map out the assignment flow on paper."*

P8: *"[I] make bulletin points on paper of what I'm trying to achieve."*

P13: *"[I] read through specs and diagram out a plan - sometimes there's psuedocode [sic] involved."*

P14: *"[I] first write out all the key points from the assignment. Then [I] do a detailed plan of what I need to code."*

Another method cited (5 of 15) was forming a mental plan. Two of the five people that cited making a mental plan also cited using written planning.

P4: *"Before I start, I dissect it [the assignment] into pieces and think about how to solve each part. Once I've thought about possible solutions, I think about possible problems."*

P6: *"[I] break the requirements into a mental hierarchy."*

The students developing a mental plan were following an identical process to the students that choose to use paper and pencil. They were outlining ideas and constructing a plan. The main difference was their choice to not write anything down. The other two actions students mentioned that they did at the beginning of an assignment were reviewing the assignment's specifications (5 of 15) and doing research (2 of 15).

3.4.1.6. *How students resume.* We asked students about starting and resuming an assignment to see if there was any difference between the two actions. We found students gave different answers in each case. When resuming a programming assignment, the most frequent answers were that the students reviewed code (7 of 15) or they reviewed notes (6 of 15) to help recover context. Four students cited both techniques in their answers.

P5: *"[I] review previously written code, look at any notes, and attempt to pick up where I left off."*

P8: *"[I] look over my bulletin points, then read the last code I wrote."*

P13: *"[I] read through code I had written in the previous session to get back to what was going on beforehand."*

The content of this feedback and its frequency is similar with the resumption strategy Parnin [20] labeled 'Return to last method modified, and navigate to related code to jog your memory', which developers mentioned 58% of time is his study.

Some students (4 of 15) said that they needed to figure out where to go and what to do next. This is explicit feedback that highlights the problem that memory cues and resumption strategies attempt to solve. While this feedback is not mutually exclusive from the above feedback, because the act of resuming requires regaining context, it is noteworthy for being explicit.

P10: *"[I] recap the last session's thought process and try to understand where to go."*

Lastly, a couple students (2 of 15) stated they reviewed the assignment's requirements and/or instructions to help regain context.

3.4.2. Follow up Questionnaire. The six continuous rating scale questions and their answers are summarized in Table 3.2. Again, 0 implied 'strongly disagree' and 10 implied 'strongly agree'.

We found that students agreed with all statements. More strongly, three of the statements had mean scores above 8. Part two's questions focused on the seminar's effect and answers are summarized in Table 3.3.

When answering 'yes', participants were asked to elaborate and provide an example. For the first question, here is a sample of what the students said:

P15: *"Yes, it [the seminar] changed the way I prepare to finish; I left more comments, concerns, and on purpose errors. It gave me a faster way of fixing the issues though."*

P4: *"It makes me think about how to walk away from code in the best way."*

Table 3.2. Pilot study: Summary of answers to part one of the follow up questionnaire.

Statement	**Mean**	**SD**
I found the information in the memory cues seminar to be useful.	7.66	1.07
I found the information in the memory cues seminar to be useful during my last programming assignment.	6.83	1.68
I found the information in the memory cues seminar to be useful for something other than my last programming assignment.	6.64	1.99
I will consider using memory cues and resumption strategies when I program.	8.57	0.84
I think memory cues and resumption strategies should be presented to CS students at some point of their curriculum.	8.33	1.43
I am interested in learning more about memory cues, resumption strategies, or similar topics.	8.19	1.34

Table 3.3. Pilot study: Summary of answers to part two of the follow up questionnaire.

Question	**Yes**	**No**
Did the seminar change the way you think about programming?	73%	27%
Did you find anything useful about the seminar?	100%	0%
Did you find anything not useful about the seminar?	0%	100%
Did you do anything differently during the last programming assignment?	80%	20%

P5: *"I've been trying to force myself to leave off with notes, etc. to help start again later."*

P7: *"[...] it made me more aware of time needed to get refocused on the programming project."*

P12: *"It caused me to make a conscious choice to leave those queues [sic] for myself."*

P13: *"It made me consider how much interruptions really affected my programming flow."*

All students stated the seminar was useful and offered their comments on how it was useful. Students mentioned that the seminar increased their awareness about the subject of memory cues and resumption strategies.

P5: *'It was good to finally formalize many of the concepts I had sort of halfway taught myself before."*

P1: *"Realizing things that I do that are already part of memory cues was helpful because it makes it easier to do them in the future."*

P3: *"Just being aware of it [the seminar's content] is huge."*

P12: *"It made me more aware of memory cues that I could use."*

P13: *"[It] made me consider more about remembering what's going on when I'm programming instead of just getting up and doing something else."*

Other comments touched on the seminar increasing the students' efficiency.

P4: *"I've gotten better at getting in the zone after a break."*

P6: *"It helped me learn how to pick up programming after a break faster."*

P9: *"'It brought some techniques to my attention that were useful to me during my subsequent assignments."*

P8: *"[...][I tell] people to wait when they interrupt me."*

Conversely, there were no examples of students saying that the seminar was not useful. When asked about how the seminar affected their approach to their last programming assignment, some of the comments (6 of 15) centered on more documentation and note taking.

P5: *"I mainly took extra notes and focused on useful documentation."*

P13: *"[I] documented my code and ideas a heckuva lot more!"*

P2: *"[I] left myself comments when I took breaks."*

P6: *"I made mental notes before breaking."*

P14: *"I commented a lot more and left myself notes from one session to another."*

Another group of respondents (3 of 15) cited thinking more about when to stop working.

P4: *"I put an effort into leaving off at a point that is easy to come back to."*

P12: *"I was more deliberate in leaving a place to restart quickly from."*

This suggested some students developed a new suspension strategy or altered an existing one.

3.5. Discussion

Going into the pilot study, our participants had little or no knowledge about memory cues and resumption strategies. Every student said the seminar contained useful information, and four out of five students changed how they worked because of the information they learned. Furthermore, students showed interest in learning more and thought memory cues and resumption strategies should be taught in a computer science curriculum. Having significant student interest in learning about this material is a compelling argument to consider. This can be used to start conversations among computer science educators about adding memory cues and resumption strategies to their classes. It also supports the idea that programming is a process that has the characteristics of being interruptible and incurring resumption lag. Recognizing these characteristics and knowing techniques to increase productivity under these conditions is a valuable skill to programmers. Not only did our participants affirm the usefulness of this information, they also applied it immediately in their subsequent programming assignment.

Furthermore, the seminar's information was even considered useful outside of the context of their last assignment. In fact, the responses about the seminar's usefulness inside (M:

6.83, SD: 1.68) and outside (M: 6.64, SD: 1.99) the context of their last assignment were indistinguishable ($p > 0.77$). We believe the likely explanation for this result is that when students refer to other work, they in part, are referring to other computer science classes that require programming. Determining if the seminar was useful to students in non-programming classes would be explored in the subsequent user study.

While many students already intuitively utilized memory cues to some degree, typically in the form of code comments, the seminar reinforced those habits and encouraged students to take more steps to help reduce their resumption lag. This point highlights the difference between implicit and explicit teaching. Without being taught, students had developed resumption strategies over the course of their learning, however their comments describe a noticeable difference after they had learned about the topic explicitly in the seminar. This supports the idea that interruptions, memory cues, and resumption strategies are all topics that can treated like writer's block and be discussed in class by teachers to directly speak to the problems students face during their creative process.

Respondents generally provided fair amounts of information on the questionnaires as evidenced by their feedback. Also, they were consistent with their positions. For example, in the initial survey, the average length of time of a productive sitting (M: 2.98, SD: 1.18) and the time a student is able to stay in the zone (M: 2.87, SD: 1.30) were indistinguishable ($p > 0.81$).

Also, as mentioned briefly in the findings section, some of our findings supported the results of previous research. In particular, we found our students' responses to the first survey to regarding resumptions and suspensions paralleled work by Parnin [21]. Their average resumption lag was 12.3 minutes, which is consistent with the fifteen minutes cited by DeMarco [29] and Solingen [27]. These preliminary results suggested that a broader user study would be a worthwhile endeavor and allow us to establish more confident results.

CHAPTER 4

User Study

The results of our pilot study motivated us to conduct a user study. In particular, we wanted to see if our pilot study's results would hold across a larger and more diverse participant pool. It is possible that pilot study's participants were not representative of the actual student population. In addition, we wanted to lengthen the time between seminar and follow-up surveys in order to lessen the likelihood the seminar's material was fresh in students' minds. Our research goal was to examine the effects on college students of presenting them instruction on interruptions, memory cues, and resumption strategies. More specifically, our objective was to support the following hypotheses:

H1: College students are affected by interruptions and have the same resumption lag as industry software engineers.

H2: College students' knowledge of memory cues and resumption strategies is limited.

H3: College students react favorably to being taught memory cues and resumption strategies in CS courses and believe they should be included in a college curriculum.

- **H3a:** The student's age will affect the student's response to being taught these topics.
- **H3b:** The student's programming experience will affect the student's response to being taught these topics.
- **H3c:** The student's years in college will affect the student's response to being taught these topics.
- **H3d:** The student's previous knowledge of memory cues and resumption strategies will affect the student's response to being taught these topics.

H3e: College students will report that learning memory cues and resumption strategies benefited them outside of CS courses.

To accomplish this, the study had three components: an initial survey, a seminar on memory cues and resumption strategies, and a follow-up survey. The study involved 198 undergraduate CS students attending a public university. Their ages ranged from 18 to 39, their experience in programming ranged from a few months to ten years, and their college experience ranged from one semester to eight years. They were not compensated for participating.

The surveys and seminar were given to five different computer courses: one freshman-level course, one sophomore-level course, one junior-level course, and two senior-level courses. The first survey and seminar were given in the middle of the semester. The follow-up survey was given at the end of the semester, roughly six weeks later.

4.1. The initial questionnaire

The first questionnaire was given in the middle of the semester and consisted of 18 questions, which was slightly longer than the initial questionnaire from the pilot study. Students completed the survey in 5 to 10 minutes. The survey gathered various types of data: background information on the student (3 questions), how their time was spent completing an assignment (6 questions), how quickly they were able to focus (3 questions), what their resumption habits were (3 questions), and if they had prior knowledge of memory cues and resumption strategies (3 questions). Once the questionnaires were completed, the seminar started.

4.2. The seminar on memory cues and resumption strategies

Seven seminars were held during seven different lecture periods in five different courses. The content was identical to that of the pilot study and each seminar lasted the same amount of time as before. The only difference was that seminar size ranged from about 15 to 100 students, depending on the lecture.

4.3. The follow-up questionnaire

The second survey was given six weeks after the seminar. The survey was a modified version of the one given in the pilot study. As before, the survey consisted of two parts. Part one was comprised of six continuous rating scale questions. The first three questions had "highly not useful" on the left and "highly useful" on the right. The final three questions had "strongly disagree" on the left and "strongly agree" to the right. The statements centered on the seminar's effect on students. More details can be found in the subsequent Results section. The second part of the questionnaire asked for details about the effect of the seminar. It also contained two questions which were identical to questions from the initial questionnaire and asked students about their start and resumption lag times. Space was also provided to allow participants to leave additional comments.

4.4. Results

The goal of this section is to support or reject the hypotheses defined in Section 4 by presenting the results of the two questionnaires.

4.4.1. H1 - College students are affected by interruptions and have the same resumption lag as industry software engineers.

Students reported a mean resumption lag of 15.12 minutes (SD: 12.72). This is consistent with the fifteen minutes cited by industry software engineers in studies by DeMarco and Lister [29] and Solingen *et al.* [27]. Students were also asked what caused a session of work to be productive or unproductive. Table 4.1 shows a coalesced summary of student responses.

TABLE 4.1. User study: Top contributors to productive/unproductive work for CS students.

Student Response	%
distractions	25%
understanding/clear requirements	24%
state of mind/mood	10%
focus	9%
time	6%
fatigue	5%

The two most mentioned factors that affected productivity were distractions (25%) and understanding/clear requirements (24%). Distractions were primarily related to environment, which included noise, other people, and non-work technology. These factors were described by students in their feedback.

P76: *"Productive: no distraction from classmates/social media."*

P85: *"[affected by] distractions/loud talking in the computer lab."*

P119: *"If I don't have interruptions, I will be productive."*

As before with the pilot study, we were encouraged by the comments on distractions because in many of the examples cited, the distraction was synonymous with a type of interruption and the seminar would directly deal with this topic. This provided evidence that CS students are affected by interruptions while doing their assignments. Again, while this was expected, we felt it was important to have supporting feedback from students.

4.4.2. H2 - College students' knowledge of memory cues and resumption strategies is limited. Nearly three-fourths (73%) of students reported not knowing memory cues or resumption strategies. Of the 27% that heard of these techniques, 42% said they

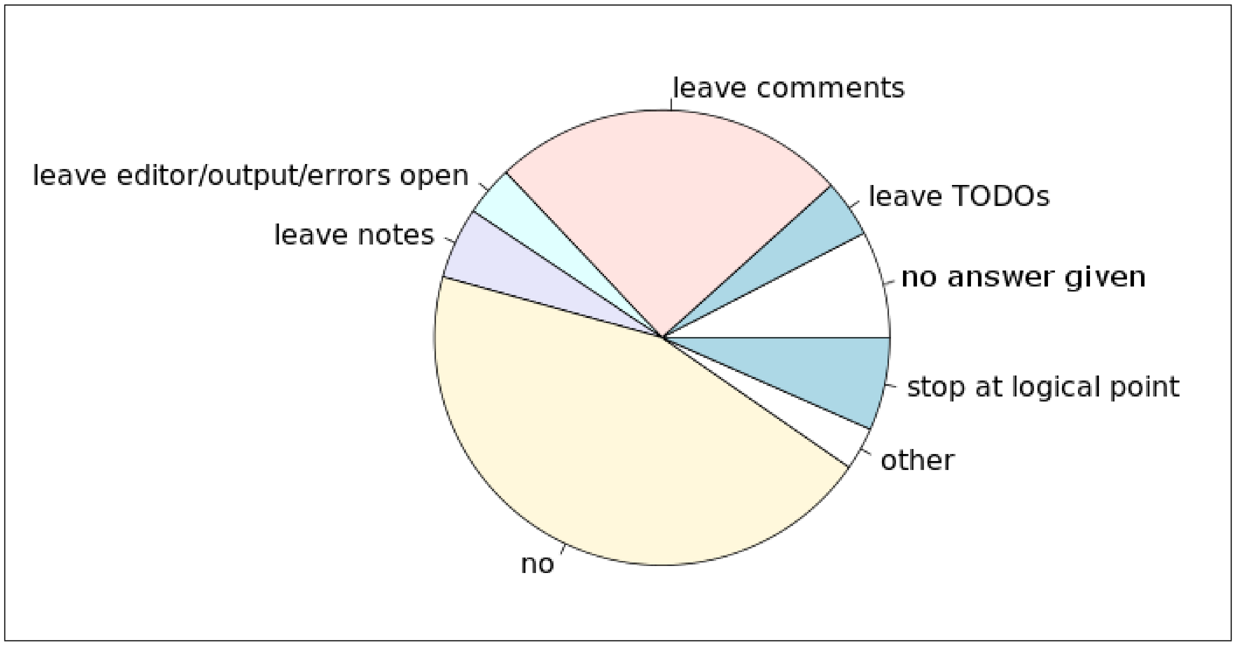

FIGURE 4.1. User study: Student responses to the question *"Do you do anything prior to stepping away from an assignment that helps you refocus when you return?"*

learned about them in a psychology class and 14% did not remember where they learned about them. No student cited a CS class as a source for learning memory cues.

We also asked if they did anything prior to stepping away from an assignment that helped them refocus more quickly when they returned to work, which could indicate if they implicitly used memory cues and resumption strategies. Figure 4.1 shows a pie chart of all answers. The most common response (48%) was "no". The second most common response (27%) was that they left comments for themselves.

4.4.3. H3 - College students react favorably to being taught memory cues and resumption strategies in CS courses and believe they should be included in a college curriculum.

Part one of the follow-up questionnaire contained six continuous rating scale questions. For the first three questions (FQ1-3), 0 implied "highly not useful" and 10 implied "highly useful", and for the last three (FQ4-6), 0 implied "strongly disagree" and 10 implied "strongly agree". Table 4.2 contains a summary of student answers and Figure 4.2 displays a boxplot for their answers.

Table 4.2. User study: Summary of answers to part one of the follow-up questionnaire.

Statement	**Mean**	**Median**	**SD**
I found the information in the memory cues seminar to be... (FQ1)	6.24	6.50	2.04
For my programming assignments, I found the information in the memory cues seminar to be... (FQ2)	6.27	6.5	2.24
For my non-CS assignments, I found the information in the memory cues seminar to be... (FQ3)	4.86	5.00	2.28
I will consider using memory cues and resumption strategies when I program. (FQ4)	7.41	7.75	2.02
I think memory cues and resumption strategies should be presented to CS students at some point of their curriculum. (FQ5)	7.78	8.00	1.93
I am interested in learning more about memory cues, resumption strategies, or similar topics. (FQ6)	6.34	6.53	2.40

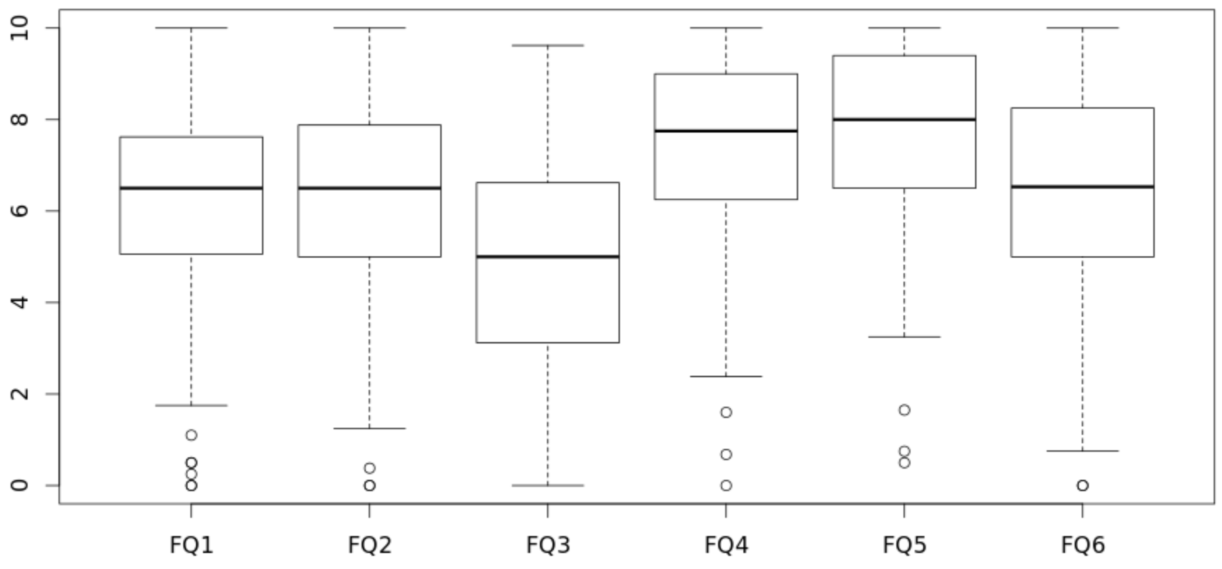

Figure 4.2. User study: Blox plot of follow-up answers

Students found the seminar useful (M: 6.24 SD: 2.04) and applicable to their programming assignments (M: 6.27 SD: 2.24). Students elaborated on these positions with their comments:

P17: *"I sometimes leave an assignment without finishing it, so it is helpful to create ways to jumpstart the work when I return."*

P18: *"[The techniques I learned] helped me reduce the time it took to get back into working after a break or interruption."*

More strongly, students said they would consider using the techniques (M: 7.41, SD: 2.02) and were highly in favor of teaching memory cues (M: 7.78, SD: 1.93). As one student articulated in his feedback:

P163: *"Seriously, try to get this stuff embedded in the CS courses. It is very important to be aware of this stuff."*

Also, they wanted to learn more about the subject (M: 6.34, SD: 2.40). For example, one student wrote:

P176: *"I'd like to know more about this topic; [I] wish it was covered in some CS course."*

Part two of the follow-up questionnaire focused on the seminar's effect and its answers are summarized in Table 4.3.

TABLE 4.3. User study: Summary of answers to part two of the follow-up questionnaire.

Question	**Yes**	**No**
Did the seminar change the way you think about programming?	57%	43%
Did you do anything differently during recent programming assignments?	43%	57%

The 57% of students that cited a difference in thinking were asked to elaborate and provide an example. Here is a sample of what the students said:

P126: *"I always write a note to myself in my program now when I find a stopping place. This helps me resume my original thought process and helps the way I think about programming."*

P95: *"[The seminar] made me aware of the pitfalls of memory and productivity."*

While 57% of students stated they did nothing differently in their recent assignments, 5% of students mentioned their assignments were too short and/or easy to warrant using the cues and recognized their usefulness in the future with more complex assignments. As one student reported:

> **P66:** *"It's [the seminar's topics] good stuff but I didn't use it this semester because the assignments weren't that hard. I'm sure I will next semester."*

Exploring the sub-hypotheses (H3a-H3e), Welch Two Sample *t*-tests were conducted on the six continuous rating scale questions (FQ1-FQ6) to determine if there was a difference between students that reported prior knowledge of memory cues and resumption strategies and students that did not report prior knowledge. Tests on FQ1 ($p < 0.05$), FQ2 ($p < 0.01$), and FQ4 ($p < 0.10$) showed statistically significant differences, which provided some support to H3d.

However, not all hypotheses were supported. Students did not report a noticeable benefit to the seminar outside of CS courses (H3e) (M: 4.86, SD: 2.28). Analysis of variance was conducted to examine the effect of age (H3a), years in college (H3c), and programming experience (H3b) on FQ1-FQ6. The results are summarized in Table 4.4. Results showed no significant difference among different values of the dependent variable. The only exception was an effect of programming experience on FQ5 ($p < 0.05$). However, post-hoc analysis with Tukey's HSD correction showed no distinguishable differences among different values of the dependent variable. Hence, our study did not demonstrate results that supported H3a, H3b, H3c, or H3e.

TABLE 4.4. User study: Results of Analysis of Variance tests on the quantitative self-reported responses from the follow-up questionnaire.

Question	Group	$\boldsymbol{F}$-statistic	$\boldsymbol{p}$-value
FQ1	Age	$F(16, 135) = 0.71$	$p = 0.78$
	College Exp.	$F(6, 145) = 1.17$	$p = 0.33$
	Prog. Exp.	$F(5, 147) = 0.60$	$p = 0.67$
FQ2	Age	$F(16, 135) = 1.22$	$p = 0.27$
	College Exp.	$F(6, 145) = 1.42$	$p = 0.21$
	Prog. Exp.	$F(5, 147) = 1.93$	$p = 0.11$
FQ3	Age	$F(16, 135) = 0.91$	$p = 0.55$
	College Exp.	$F(6, 145) = 0.49$	$p = 0.82$
	Prog. Exp.	$F(5, 147) = 0.51$	$p = 0.73$
FQ4	Age	$F(16, 135) = 0.75$	$p = 0.74$
	College Exp.	$F(6, 145) = 1.49$	$p = 0.19$
	Prog. Exp.	$F(5, 147) = 0.79$	$p = 0.54$
FQ5	Age	$F(16, 135) = 1.51$	$p = 0.10$
	College Exp.	$F(6, 145) = 1.37$	$p = 0.23$
	Prog. Exp.	$F(5, 147) = 2.45$	$p = 0.05$
FQ6	Age	$F(16, 135) = 1.13$	$p = 0.34$
	College Exp.	$F(6, 145) = 1.64$	$p = 0.14$
	Prog. Exp.	$F(5, 147) = 1.97$	$p = 0.10$

4.5. Discussion

By showing that college students are affected by interruptions in similar ways to industry software engineers (H1), we were able to extend findings of previous research [29][27] and reproduce the result from our pilot study. Students' average resumption lag was 15.12 minutes, which is consistent with the fifteen minutes industry software engineers reported.

In addition, we believe the feedback from the 198 students was meaningful and consistent. Respondents generally provided fair amounts of information on the questionnaires as evidenced by their feedback. Consistency was demonstrated by their answers. For example, in the initial survey, we included two questions that were intended to ask the same thing and only differ in wording. In one question, we asked what was the average length of time of a productive sitting for the student. In the other, we asked how long the student was able to "stay in the zone". We performed a Welch Two Sample t-test on the results of the former (M: 2.40, SD: 1.34) and the latter (M: 2.60, SD: 3.43) and found they were indistinguishable ($p > 0.50$).

Students of all ages and experiences are of the strong opinion (M: 7.78, SD: 1.93) that this material should be part of their undergraduate CS curriculum. Furthermore, 57% changed the way they thought about programming as a result of the seminar and 43% did things differently. Additionally, a few more mentioned they would have done something new if their assignments were more challenging. These results tell us that college students react favorably to being taught memory cues and resumption strategies in CS courses and believe they should be included in a college curriculum (H3).

We expected the student's age, years in college, and programming experience to affect the student's response to being taught memory cues and resumption strategies (H3a, H3b, H3c). However, none of these factors affected students' responses significantly. The significant difference was between students with and without prior knowledge of memory cues and resumption strategies (H3d), where the quarter of students with prior knowledge finding the seminar less useful, yet they agreed that the material should be taught.

We also thought students would report a benefit to learning about memory cues and resumption strategies that extended to non-CS courses (H3e), because our pilot study's participants reported a benefit. However, our results showed that students were neutral, at best, to that statement.

While other disciplines, namely Psychology, touch on memory cues and resumption strategies, CS teachers in universities cannot rely on students being exposed to these subjects, as shown by the results to H2. Therefore, there is a clear knowledge gap that many

CS students have. And, according to their feedback (H3), they desire to fill this gap and learn more.

We believe another takeaway is that while roughly half of students already intuitively utilized memory cues to some degree, typically in the form of code comments, the seminar reinforced those habits and encouraged students to take more steps to help reduce their resumption lag. As one student wrote,

> **P181:** *"I've been doing something similar [to what was taught in the seminar], but this made it more concrete. I'd like to learn more."*

Finally, we noted that the pilot study yielded higher self-reported feedback than the user study. We believe this is because the pilot study's participants actually came to us to volunteer. Hence, we think this made them more likely to be interested students and more apt to react favorably to the new material.

CHAPTER 5

Limitations

In this section, we remark on limitations and possible counterarguments to our work. First, we have not established to what extent memory cues and resumption strategies should be taught, only that students believed that they should be taught. Readers can use the description of the seminar in the method section to see what material was covered, but this is not intended to be a definitive guide of what CS teachers should adopt. Again, our objective was to show that teaching students about interruptions, memory cues, and resumption strategies is welcomed by students, not how it should be done in the classroom.

Secondly, our participants came from the same university. However, our institution is similar to other public state universities. While we believe there was sufficient diversity among our participants, one could conduct a second study across multiple education institutions.

Thirdly, students might have been biased to provide positive feedback because they interpreted positive feedback as being helpful. This occurred to the authors at the outset of the study, and clear verbal instructions were emphasized to the participants to be honest and that there were no right answers. Additionally, the course instructors reiterated to their classes that no favoritism or reward would be given to those who participated in the study.

CHAPTER 6

Future work

A natural progression of this research would be to establish what topics are the most valuable to cover with respect to interruptions, memory cues, and resumption strategies. While our seminar was well-received by students, we have no way of knowing if the seminar could be improved and to what extent material should be covered. Hence, we envision a new study to determine which topics are most valuable to share with students and would be a logical next step in research.

We also believe there is a vast potential for research into “programming as a craft” that could take existing knowledge and research of the creative process and cognitive psychology and apply it to programming. We believe knowledge from these areas that is introduced into CS can only enhance the field and improve the education of its students.

CHAPTER 7

Conclusion

Our environment is filled with distractions and glancing around us appears to suggest the number of distractions will only grow. Mobile devices are currently becoming more and more ubiquitous in our world and enjoy vibrating and ringing to attract our attention to the latest email, news story, or social media message. Students are growing up in a world where technology actively notifies them of updates at seemingly instantaneous rates. Those that enter industry and work as software engineers will often be required to juggle their programming workload with a plethora of team meetings, manager one-on-ones, and co-worker questions. Giving students mechanisms to cope with these distractions was one of the motivations for exploring the topic of educating students about interruptions and how to mitigate their effect.

Therefore we examined if students were affected by interruptions, what knowledge students possessed regarding memory cues and resumption strategies, and what their opinion was of learning this material. We found that students were affected by interruptions and cited a mean resumption lag of 15 minutes which extended findings from previous research on professional software engineers. In addition, nearly three-fourths of students reported not knowing about methods to mitigate the effects of interruptions. After learning about memory cues and resumption strategies, students reported that the material was useful and that they wanted to learn about them. Their most significant feedback was that they had a strong desire to include these techniques in CS curriculums. Furthermore, their feedback, including their interest in learning more and their opinion that it is a worthwhile topic to be covered in class, was independent of their age, college experience, and work experience. Based on our findings, we hope this research will open the door to more studies of teaching students about the creative process. Thus, much like English students, their knowledge would extend beyond the technical details of their craft and therefore they would be equipped to deal with the distractions our evolving environment throws at them on a daily basis.

BIBLIOGRAPHY

[1] A. M. Szóstek and P. Markopoulos, "Factors defining face-to-face interruptions in the office environment," in *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, (New York, NY, USA), pp. 1379–1384, ACM, 2006.

[2] D. McFarlane, "Coordinating the interruption of people in human-computer interaction," in *Human-computer interaction, INTERACT* (A. Sasse and C. Johnson, eds.), vol. 99, p. 295, 1999.

[3] D. C. McFarlane and K. A. Latorella, "The scope and importance of human interruption in human-computer interaction design," *Hum.-Comput. Interact.*, vol. 17, pp. 1–61, Mar. 2002.

[4] P. D. Adamczyk and B. P. Bailey, "If not now, when?: The effects of interruption at different moments within task execution," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, (New York, NY, USA), pp. 271–278, ACM, 2004.

[5] H. M. Hodgetts and D. M. Jones, "Reminders, alerts and pop-ups: The cost of computer-initiated interruptions," in *Proceedings of the 12th International Conference on Human-computer Interaction: Interaction Design and Usability*, HCI'07, (Berlin, Heidelberg), pp. 818–826, Springer-Verlag, 2007.

[6] H. M. Hodgetts and D. M. Jones, "Resuming an interrupted task: Activation and decay in goal memory," in *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci 2006)*, p. 2506, 2006.

[7] T. J. Robertson, S. Prabhakararao, M. Burnett, C. Cook, J. R. Ruthruff, L. Beckwith, and A. Phalgune, "Impact of interruption style on end-user debugging," in *Proceedings*

of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, (New York, NY, USA), pp. 287–294, ACM, 2004.

[8] D. McFarlane, "Comparison of four primary methods for coordinating the interruption of people in human-computer interaction," *Hum.-Comput. Interact.*, vol. 17, pp. 63–139, Mar. 2002.

[9] S. T. Iqbal and B. P. Bailey, "Leveraging characteristics of task structure to predict the cost of interruption," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), pp. 741–750, ACM, 2006.

[10] D. D. Salvucci, N. A. Taatgen, and J. P. Borst, "Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1819–1828, ACM, 2009.

[11] L. Marulanda-Carter and T. W. Jackson, "Effects of e-mail addiction and interruptions on employees," *Journal of Systems and Information Technology*, vol. 14, no. 1, pp. 82–94, 2012.

[12] M. Czerwinski, E. Horvitz, and S. Wilhite, "A diary study of task switching and interruptions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, (New York, NY, USA), pp. 175–182, ACM, 2004.

[13] R. Harr and V. Kaptelinin, "Unpacking the social dimension of external interruptions," in *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, GROUP '07, (New York, NY, USA), pp. 399–408, ACM, 2007.

[14] R. Harr and V. Kaptelinin, "Interrupting or not: Exploring the effect of social context on interrupters' decision making," in *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, (New York, NY, USA), pp. 707–710, ACM, 2012.

[15] Q. R. Jett and J. M. George, "Work interrupted: A closer look at the role of interruptions in organizational life," *Academy of Management Review*, vol. 28, no. 3, pp. 494–507, 2003.

[16] J. Jin and L. A. Dabbish, "Self-interruption on the computer: A typology of discretionary task interleaving," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1799–1808, ACM, 2009.

[17] E. M. Altmann and J. G. Trafton, "Timecourse of recovery from task interruption: Data and a model," *Psychonomic Bulletin & Review*, vol. 14, no. 6, pp. 1079–1084, 2007.

[18] T. Jackson, R. Dawson, and D. Wilson, "Reducing the effect of email interruptions on employees," *Int. J. Inf. Manag.*, vol. 23, pp. 55–65, Feb. 2003.

[19] J. L. Franke, J. J. Daniels, and D. C. McFarlane, "Recovering context after interruption," in *Proceedings 24th Annual Meeting of the Cognitive Science Society (CogSci 2002)*, pp. 310–315, 2002.

[20] C. Parnin and R. DeLine, "Evaluating cues for resuming interrupted programming tasks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 93–102, ACM, 2010.

[21] C. Parnin and S. Rugaber, "Resumption strategies for interrupted programming tasks," *Software Quality Control*, vol. 19, pp. 5–34, Mar. 2011.

[22] L. Kara A., "Effects of modality on interrupted flight deck performance: Implications for data link," tech. rep., 1998.

[23] S. Gievska, R. Lindeman, and J. Sibert, "Examining the qualitative gains of mediating human interruptions during hci," in *Proceedings 11th International Conference on Human-Computer Interaction, Las Vegas, Nevada*, 2005.

[24] N. Kern, S. Antifakos, B. Schiele, and A. Schwaninger, "A model for human interruptability: Experimental evaluation and automatic estimation from wearable sensors," in *Proceedings of the Eighth International Symposium on Wearable Computers*, ISWC '04, (Washington, DC, USA), pp. 158–165, IEEE Computer Society, 2004.

[25] P. Palanque, M. Winckler, J.-F. Ladry, M. H. ter Beek, G. Faconti, and M. Massink, "A formal approach supporting the comparative predictive assessment of the interruption-tolerance of interactive systems," in *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '09, (New York, NY, USA), pp. 211–220, ACM, 2009.

[26] J. Chong and R. Siino, "Interruptions on software teams: A comparison of paired and solo programmers," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, (New York, NY, USA), pp. 29–38, ACM, 2006.

[27] R. van Solingen, E. Berghout, and F. van Latum, "Interrupts: Just a minute never is," *IEEE Softw.*, vol. 15, pp. 97–103, Sept. 1998.

[28] E. Hemingway, G. Seldes, T. Dreiser, L. Hughes, and A. D. Ficke, *Monologue to the maestro: A high seas letter*. Esquire, 1935.

[29] T. DeMarco and T. Lister, "Peopleware: Productive projects and teams dorset house," *New York*, 1999.