Understanding Students' Frustration and Confusion during a Programming Task: A Multimodal Approach

Rakhi Batra
Department of Computer Science and Engineering
The Ohio State University
Columbus, USA
rakhi.1@osu.edu

Zahra Atiq
Department of Computer Science and Engineering
The Ohio State University
Columbus, USA
atiq.2@osu.edu

***Abstract*— For novice students, learning programming is hard, hence, inducing a variety of emotions. According to literature, two of the most commonly occurring emotions that students experience while learning programming are frustration and confusion. Although these emotions are momentary, they may have long-term effects on students' motivation, performance, and retention in computing. In this study, we aim to discuss challenges that students report when they feel frustration and confusion while working on the rainfall problem. This problem has been used extensively in literature to understand students' problem-solving skills. However, little is understood about how students react emotionally when they work on this problem.**

We recruited twenty-eight students who took CS1 during Fall 2022. They worked on the problem for twenty minutes while we collected their biometrics, clickstream, and keystroke data. A retrospective think-aloud interview was conducted soon after the task, where participants elaborated on their emotional experiences while watching the video replay of their programming task. We analyzed interview data using qualitative content analysis and triangulated these findings with biometric, clickstream, and keystroke data. Some of the challenges that trigger confusion and frustration are getting unexpected output, inability to resolve compilation and logical issues, forgetting syntax, and conceptual misunderstanding. Moreover, we found an alignment between the different data sources to provide a near real-time view of emotional experiences. Instructors may use the findings of this study to design interventions that support problem-solving, such as problem-based teaching, using interactive programming tools, and tracing for debugging.

Keywords—confusion, frustration, multimodal data, rainfall problem, biometrics, triangulation, engineering education

I. INTRODUCTION

Programming is a fundamental skill to learn for computing students, but it is often challenging because it involves learning abstract concepts, programming syntax, and complex problem-solving techniques [1]. Considering the difficulty of learning programming, a major focus is dedicated to improving teaching practices, for instance, a computerized learning environment that captures student behaviors and provides customized feedback [2]. However, these environments somewhat neglect the emotions that students experience while learning programming [3]. Students experience a variety of emotions throughout their learning and different emotions impact differently on their performance and motivation [4]. Literature suggests that emotions, such as confusion and frustration, occur frequently during computer programming sessions [3, 5-8] and these states are correlated with student performance [9]. Moreover, these two emotions have mixed effects on learning outcomes. Some studies report a positive effect of confusion and frustration on learning [10, 11] whereas other studies report a negative effect [9]. For instance, [8] found that brief confusion or frustration results in positive outcomes and enhances learning whereas longer confusion leads to frustration, which hinders learning and affects the outcome negatively. Similarly, [12] reported that if the positive state of confusion is not resolved, it transitions to frustration, and ultimately to boredom. Considering the role of these emotions in learning, the purpose of this study is to understand students' confusion and frustration while doing programming. Specifically, we aim to answer this research question: *"What challenges do students report when they feel frustration and confusion while working on the rainfall problem?"* For the purpose of answering our research question, we used a multimodal approach for data collection. Multimodal data helps capture multi-layered information about the complexity of human emotions from different perspectives [13]. The data analysis methods are primarily qualitative followed by the description of two exemplars that explain the triangulation between the different modalities.

II. LITERATURE REVIEW

A. Conceptual Framework

The control-value theory (CVT) describes multiple types of academic emotions. Two types of emotions explained in CVT are achievement emotions and epistemic emotions. The terms "affect" and "emotion" are sometimes used interchangeably in the literature. However, these two are different. Affect refers to the "basic sense of feeling ranging from unpleasant to pleasant and idle to activated", while emotions are intense and short-term usually triggered by some event [14].

Frustration is an activity-related achievement emotion. It triggers as a result of failure in an activity that is not sufficiently controllable [15]. For instance, during programming labs, students feel frustrated when they are not able to identify logical errors in their code, or if they get incorrect output.

Epistemic emotions, on the other hand, emerge as the result of solving cognitively demanding tasks. These tasks usually hinder or aid the learning process [16]. Confusion is an epistemic emotion because it triggers after cognitive disequilibrium has occurred while working on a learning task [10, 16, 17]. Confusion could either be beneficial or detrimental to the learning process, depending on how it is induced and resolved [10]. It is important to note that CVT

2023 IEEE Frontiers in Education Conference (FIE) | 979-8-3503-3642-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/FIE58773.2023.10343499

does not define confusion as emotion, however, [18] suggests that confusion shares the properties of emotion such as identifiable facial markers and appraisal structure, which makes it enough to be in the emotions and affective science literature [16, 18, 19].

Confusion turning to frustration and vice versa is explained by Bosch and D'Mello in their theoretical model of the affect transitions [12]. According to this model, when students encounter an impasse during a learning activity, they experience confusion. If the impasse is not resolved, this confusion triggers frustration. Moreover, frustration can transition back to confusion if students encounter new impasses, or transition into boredom if frustration persists.

B. Confusion and Frustration in Learning Programming

In the past several years, researchers have shown interest in understanding affect and academic emotions in learning [5, 8]. However, a few researchers have a specific focus on confusion and frustration because these two emotions have positive as well as negative effects on learning outcomes [9, 10, 11]. Literature suggests that there is no direct method to resolve the confusion, however, it requires students to have skills and knowledge to resolve it or the instructor can provide scaffolding to help students resolve their confusion [10, 19]. Moreover, if confusion is not resolved and students are not able to improve their understanding, they do not learn much, become frustrated, and doubt their abilities [12, 21].

The studies on confusion and frustration have used a variety of methods to identify these emotions. The most common method is self-reporting by participants. In a series of studies by D'Mello et al., the authors have studied novice students' affect during two programming phases by analyzing their self-reported affective judgments [3, 22]. The authors found that students most frequently experience engagement, confusion, frustration, and boredom while programming. Self-reported methods for understanding emotions enhance the validity of results but these methods are limited in their applicability [23]. To understand emotions as students work on a program, Rodrigo et al. used students' compilation logs from the integrated development environment (IDE) [24]. They found that a coarse-grained level of frustration can be identified by analyzing the average time between the number of errors, compilations, and the number of consecutive compilations after modifying the same code or getting the same error. However, they suggested that short-time compilation logs are not enough to identify students' frustration.

Considering the importance of the identification of emotions at the appropriate time, researchers are getting interested in the automatic identification of emotions using facial features [25, 26, 27]. In a series of studies by Grafsgaard et al., facial features were used to predict confusion and frustration in the programming context [26, 27]. They used a corpus of computer mediated human-human tutoring sessions, which contained webcam video, database logs, and skin conductance. However, for the analysis, the authors only used facial expressions to build a regression model. They found that brow lowering was positively correlated with frustration. Using similar facial features as Grafsgaard et al., but a significantly different data collection mechanism, Bosch et al. used the Computer Expression Recognition Toolbox (CERT)[1] to detect the academic emotions of students while they were learning Python using a computerized learning environment [25]. The authors built the models based on facial features and found that confusion, uncertainty, and frustration were distinguished from all other emotions using the CERT. Emotions are complex and require multi-layered processing to understand their complexity [13]. For example, in a study by Lee and Sumi, students' facial features were used in combination with students' typing and compilation logs, self-reported emotions, and one action state label (which includes reading, writing, thinking, finding, unfocused, fixing, or other) to predict emotions [28, 29]. The authors trained Hidden Markov Models (HMM) on the sequences and action units of facial expressions of engagement, confusion, and frustration. Although limiting the use of their data collection method (self-report), authors found that the inclusion of self-reported action states in model training can increase the performance of the model for identifying engagement and frustration and to some extent confusion and boredom.

In real life, facial expressions are a good source of identifying emotions however, there are other physiological measures that reflect changes when a person feels certain emotions [30, 31, 32]. For instance, Pachman et al. used eye gaze data to detect confusion in puzzle-solving tasks [33]. The authors collected students' eye-gaze data during the tasks and self-reported data after the completion of tasks. During self-reporting participants retrospectively described the thoughts they had during a puzzle-solving process and reported the confusion level. The authors calculated the fixation on areas of interest and counted the instances where confusion ratings were high. The authors found that participants mostly got confused about the non-relevant areas of the problem.

From the summary of the literature, it is possible to automatically detect certain emotions by using compilation logs, typing logs, facial features, and eye-gaze data but this may compromise the validity of results because in most studies these data sources are used in isolation. The validity of results increases by incorporating multiple sources of data [34]. Hence, we are using a multimodal approach for data collection and analysis. More specifically, we collect eye-gaze, EDA, clickstream, and retrospective think-aloud interviews (self-report) to provide a fine-grained and near real-time understanding of students' confusion and frustration during a programming task.

III. Research Methodology

A. Context

The Ohio State University, located in the Midwestern United States, is a large and well-known university in the field of engineering. It enrolls full-time, residential, and traditional-aged undergraduate students in a variety of engineering disciplines. The context for this study is composed of three introductory programming courses (CS1): CSE-1222 (C++), CSE 1223 (Java), and CSE 1224 (Python). These introductory courses are three credits offered each semester to students of all majors. These courses cover introductory programming concepts such as branching statements, loops, and handling user input. These courses employ different teaching methods, including active learning, blended learning, and project-based approaches. Students attend lectures and collaborate with their peers to solve problems during lectures. Additionally, they watch online modules outside of the classroom, work on

[1] CERT tracks facial features based on the Facial Action Coding System (FACS).

interactive readings, and complete self-check questions and projects. Each week students have a lab during which they work on problems and get help and feedback from graduate teaching assistants. Each section of these courses has an enrollment of up to 160 students. The weekly lab is divided into smaller sections of 40 students each.

B. Selection Criteria

First, we considered students' who were novices because novices may experience a range of emotions while taking a programming course [35]. In the context of this study, novices are students who have no experience or limited programming experience before taking these courses, and they are not repeating the course. Second, we used purposive sampling [36] to consider the students who represent diversity regarding gender, ethnicity, and major field of study.

C. Participants

Using the selection criteria defined above, we recruited 28 students who took CS1 during Fall 2022. The course-wise breakdown of participants is CSE 1222 (3), CSE 1223 (19), and CSE 1224 (6). The gender breakdown of participants is Male (17), Female (9), and Others (2). The ethnicity breakdown of the participants is White (11), Asian (8), Black or African American (4), Hispanic or Latino (2), and Biracial (3). For confidentiality purposes, we assigned pseudonyms to all the participants. A $20 Amazon gift card was given to each participant as compensation.

D. Data Collection Methods

For data collection, we used self-report data (think-aloud interview), physiological biomarkers (electrodermal activity and eye-tracking), keystroke, and clickstream data.

1) Retrospective think-aloud interview: In the retrospective think-aloud interview, participants provide a description of their experiences of performing a task by watching the video of the task [37]. The retrospective think-aloud interview is a suitable choice when participants engage in cognitively demanding tasks because these tasks require complete attention. However, the retrospective think-aloud interview is prone to recall bias that may lead to the participants not being able to recall critical information [38]. To minimize recall bias, we overlaid the video of the participant's task with mouse movements and eye-gaze data. We also recorded participants' facial expressions during the task and presented them along with the task recording during the retrospective think-aloud interview.

2) Biometric Data: Research has shown that emotional arousal can be correlated with biometric measures, such as electrodermal activity [30]. The psychological states are influenced by the person and the task that is being performed, and in turn, may affect their actions [39]. Based on the relation between biometric and psychological states, biometric data could provide a near real-time understanding of students' emotions that they experience while doing the programming task.

Eye-related biometric features include eye movement and pupil dilation. There are multiple measures of eye movement, however, we used eye fixations (gaze position on a specific object) [40, 41]. From fixation, we can calculate fixation duration (which indicates attention required by stimulus) [42] and fixation count (which indicates areas referred to multiple times) [32]. These metrics provide insights into the attention of a person and task complexity [41]. Skin-related biometric features include skin temperature and electrodermal activity

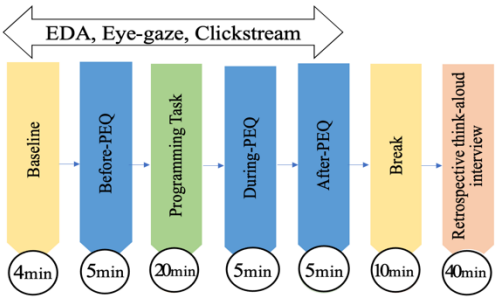


Fig. 1. Research Design

(EDA) [41]. When a person is aroused, the sweat glands in the skin produce more sweat which increases the electrical conductance of the skin. The change in electrical conductance is measured by EDA [43].

3) ClickStream and Keystroke Data: Clickstream and keystroke data represent the sequential collection of user activities or events during an interaction with a computer. It includes data such as typing something, mouse clicks and movement, and navigation paths. These data provide valuable insights into user behavior, preferences, and search patterns [44].

E. Research Design

Fig. 1 explains the research design and the order of data collection modalities within the research design.

1) Data Collection

In this study, we used iMotions software for data collection [45]. iMotions collects data related to human behavior by integrating and synchronizing different sources of data such as surveys, screen recordings, biometric sensors, and video [45]. We collected multimodal data in two sessions. In the first session, participants' biometric data, the programming task, and self-reported surveys were collected using iMotions. In the second session, we did a retrospective think-aloud interview with each participant. Details about both sessions are provided in the following sections:

First Session (Programming session): In this session, first, participants' baseline data of biometrics were collected. After that, participants filled out the survey to report their prospective emotions before the programming task and then started to work on the programming task. Participants were given twenty minutes to work on the problem. After completing the task, participants filled out the post-survey to report the retrospective emotions experienced during and after the programming task. For this study, we are not using the pre-, and post-survey data. We asked participants to choose among three programming languages and IDE to work on: Java (Eclipse), Python (VS code), and C++ (Eclipse).

Programming Task: We selected the rainfall problem as a programming task for this study, which is defined as follows: *"Write a program that repeatedly prompts the user to enter numbers (integers). Once the user enters the number 999, output the average of the non-negative numbers entered by the user, do not include 999 in the calculation of the average."*

The rainfall problem uses multiple programming constructs (such as input from the user, loops, conditional statements, and identifying division by zero error) to test students' problem-solving abilities [46]. Previous research has used the rainfall problem to understand how students plan and

compose a programming problem and considered it a challenging problem for novices [47, 48]. While working on this challenging problem, students face problem-solving difficulties [47] that could trigger negative emotions and may affect students' performance. Hence, it is important to understand the problem-solving difficulties that trigger confusion and frustration to help students overcome those difficulties.

Second Session (Retrospective Think-Aloud Interview): The second session of the study started approximately ten minutes after the completion of the first session. During this session, the researcher conducted a retrospective think-aloud interview with the students, where they replayed the screen-captured video (annotated with eye gaze data and mouse movement) of the students' programming task and paused the video every two minutes to ask students to explain their emotional experiences and actions they might take to deal with their emotions. Moreover, students were provided with a list of emotions to help describe their emotions [49].

F. Data Analysis

Directed Content Analysis (DCA): DCA is a qualitative content analysis approach that is used for the flexible analysis of text data [50]. In DCA, an existing theory or theoretical framework is used to guide qualitative data analysis. We used DCA to perform the qualitative content analysis on the interview data.

First, we extracted the definitions of emotions of our interest (confusion, frustration, confusion plus frustration) from the relevant literature. Second, we selected a process model that describes the behaviors and emotions experienced by students while programming. This process model is based on prior theory and literature on achievement emotions and CS education research [51] and is refined by the second author [5]. The six stages of the process model are getting started (GS), typing code (TC), encountering difficulties (ED), dealing with difficulties (DD), succeeding (SC), and stopping (ST). For more details on these stages and the process model refer to [5, 52].

Third, we reviewed the transcripts and highlighted all text that contained confusion and frustration. Fourth, we coded the highlighted text using the predetermined categories wherever possible. While coding we also referred to unhighlighted text as it might contain the text that aligns with the operational definitions of emotions or help in understanding the context. Fifth, we used the coded data to discuss and describe the data that supports, extends, or disapproves the theory.

Biometric Data Analysis: We used biometric data for triangulation with the qualitative findings to provide a near real-time and fine-grained understanding of students' emotions. Since we conducted the retrospective interviews in two-minute intervals, we aligned the biometric data into two-minute segments. The alignment of all data sources makes it easy to analyze data and provide a temporal snapshot of students' emotions and behaviors at a certain point in time. The biometric data were cleaned and processed using Python scripts.

IV. Findings

The findings of this study are primarily qualitative, followed by the triangulation with biometric data. The qualitative findings are categorized into three groups based on the stages in the process model: 1) Getting started, 2) Debugging (typing code, encountering difficulties, and dealing with difficulties), and 3) Ending (succeeding and stopping) [5]. Participants reported confusion and frustration frequently during the debugging stages. Therefore, we only discuss the challenges encountered in the debugging stages in the subsequent sections.

We have tried to tease out the different stages of debugging. However, it must be noted that participants may be engaged in two stages at the same time, for instance, they may be typing code while dealing with difficulties or encountering more difficulties while doing trial-and-error (dealing with difficulties).

A. Challenges that Trigger Confusion and Frustration during Debugging

While debugging, participants transition between typing code (TC), encountering difficulties (ED), and dealing with difficulties (DD) [5]. The number of transitions depends upon participants' progress with writing the code. For instance, students get stuck if they cannot fix issues in the code and repeatedly transition between TC to ED, ED to DD, or DD to TC. On the other hand, they work in flow if the code executes successfully, and they may just transition from TC to ST and then SC.

1. Typing Code

While typing code, participants were also monitoring what they were doing. During monitoring, participants sometimes get confused in selecting the appropriate programming construct for solving the programming task. For instance, Mario was anticipating that his code had something missing, as shown in the following excerpt.

"Um, the same as before, but I think my confidence started to drop 'cause I got confused between like whether I should use a while or if, or even like, a for loop 'cause I know there had to be like some nested loop; I just couldn't figure it out." - ***Mario***

Mario was confused about which construct to use. His confusion probably stems from the fact that Mario is a novice, and his conceptual understanding of programming is still in its infancy, hence he is unable to decide if he needs to use a conditional statement or a loop. He was also confused about which type of loop to use (while vs. for). Mario's experience is explained by [1] which suggests that novices typically have many deficits in programming strategies such as the way knowledge is used and applied (for instance, using an appropriate loop in a program). Moreover, Mario's confusion in using a programming construct aligns with the findings that novice students experience difficulty in using the appropriate plans while solving the rainfall problem [46].

While typing code, participants also experienced frustration when they had negative expectations about the output of the programming task. For instance, Bella explained that she overcomplicated the task and perceived that she could not finish the task in a given time although she knew how to do the task.

"Um, um, I was getting worried that I wasn't gonna be able to finish it and then I kind of was feeling a little bit bad because I was like, okay, this doesn't seem like that hard of a question. And I was probably over complicating it and so I was like feeling a little bit frustrated with myself because I probably should have been able to finish it in the 20 minutes time" -***Bella***

Bella perceived that the task as simple and doable in twenty minutes because of which she had a positive perception. However, her actual experience (not being able to complete the task and getting some errors) changed her

perception which resulted in her feeling frustrated. Bella's experience is in accordance with [46], which suggests that positive perception and negative programming episodes induce negative emotions such as frustration.

2. *Encountering Difficulties*

2.1. *When the code does not work*

The most frequent reason for experiencing confusion and frustration is when the code does not work. More specifically, when participants expected that the code would work but it did not work, they felt confused. On the other hand, unsuccessful attempts at resolving unexpected behavior of code resulted in frustration. For instance, Kevin got confused because he assumed that his code was right. However, when he executed the code, it did not work as expected.

"Um, so I basically thought I got it all right … and then after I pressed 999, it still asks for the number. And so, at this point my emotion was like, confusion.[…] Because, I mean, so the-the confidence meant that I wrote, like, wrote everything and... yeah; I felt like it was right to, like to some extent, but obviously when I looked at it, like afterwards, after I looked at the error, this is like I noticed, okay, this is a pretty major flaw. And I -- shame, shame would also be there." – ***Kevin***

Kevin's confusion aligns with the literature which states that when students encounter inconsistent or contradictory information while working on a task they feel confused [18]. Some participants got stuck when their code did not work due to syntax errors. They were not able to fix issues in their code even after spending a significant amount of time. For instance, in the following excerpt, Smith explains his emotional experience when his code did not work.

"I think I was pretty frustrated at this point 'cuz I tried it a couple times and it still wouldn't run, so I just spent most of it reading through everything to try to figure out what was wrong -- and I was pretty confused because I couldn't figure out what -- why it wasn't working." ***Smith***

Smith's emotional experience aligns with existing literature which suggests the long-term cognitive impasse changes confusion to frustration [3].

2.2. *Forgot how something worked*

Most participants experienced frustration when they forgot how something worked. This was particularly relevant for participants working in Java and they encountered challenges when they forgot how to take user input or initialize an array. For instance, in the following excerpt by Katie, she mentions that she understood the task and she could have done it in under five minutes if she had some help. For her programming class, she normally relied on notes to understand how syntax worked.

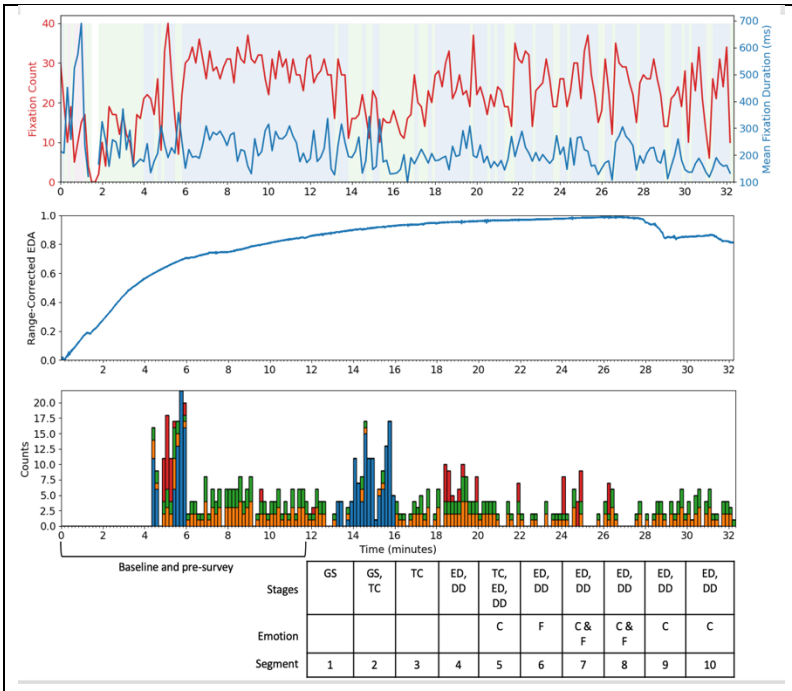
"I think I was more frustrated at this point because I felt like I knew it, and then, […] I'm just too reliant on just, previous works.[…] I was like, really frustrated at myself, […] I needed like that small little booster help to figure -- like, to finish it, […] I knew I could have finished this in like under five minutes --" ***Katie***

Katie's experience aligns with the definition of frustration by the CVT [15], which suggests that when participants have low control over the activity, failure in that activity induces frustration. During the programming task, Katie was not allowed to use outside help. She tried to use IDE-based help but could not get any useful information. Due to limited control in getting help, she could not do the task and felt frustrated.

To solve the rainfall problem, some participants tried to use an array because they considered an array to be an appropriate data structure that could store multiple inputs. The following excerpt by Brandon explains how he forgot to initialize an array.

*"A little bit more frustration: um, I was getting very frustrated. I couldn't remember the syntax, uh, to declare an array and create an arry, um, but I was still trying to analytically work my way through it." -***Brandon**

Brandon wrote code in Java where the size of the array needs to be known before creation, but in the given rainfall problem, the number of inputs is unknown, leading to an unknown array size. Literature suggests that the rainfall problem can be solved using a single variable instead of arrays [47, 54]. Brandon's decision to use an array indicates a gap in the understanding of programming constructs for novice



a

Segment 05: *it would either display like, a error message -- if something was wrong before like the average, […] or it would just straight up display the averages and then that, and then like some random number that maybe was wrong, but like still would display something.*

Segment 06: *I was just like looking it over --- and like trying to see like look for mistakes and like, I was like, trying to, um, do different things and like output things --*

Segment 07: *I figured that it was like, actually like printing out what I coded and it was like -- like just like outputting like something else, like some other program at the end, but like, I don't know, <laugh>; I might be wrong, but [..] I couldn't figure out why it wasn't like printing out like the average part, and I was like really trying to like change around and like trying to see what I was doing wrong, but like -- I don't know -- like I was very focused.*

Segment 08: *I already was like, I do not know why is not doing anything that I'm like putting in here -[..] I think I'm doing this right; like, there's something wrong with computer, not me.*

Segment 09: *this shouldn't display like over and over again 'cuz it's not in a while loop. So, this is what it should be printing out here, but it's not printing the same thing as it is here in the program, […] -I was trying to think like if I was running the program wrong or something like that.*

Segment 10: *I was confident that I did it right like then, but I was like, mostly confused why it wasn't running. Like, if I was -- like the coding part where I had written, I was -- I saw that it was right; well, at least like I did my best.*

b

Fig. 2 (a) Jane's Eye-gaze (top), EDA (middle), and Clickstream and Keystroke (bottom) graphs over the 2-minute segments and the table on the bottom shows the occurrence of confusion (C) and frustration (F) during different stages of the process model (b) Interview excerpts of segments where the participant experienced confusion and frustration.

students. His experience confirms with literature [1, 54], which suggests that novices face difficulties in solving programming problems due to semantic misinterpretation of programming constructs.

3. *Dealing with Difficulties*

One of the main ways that novices deal with difficulties is by engaging in trial and error [5]. Our data also confirm this behavior of novices. During this stage, participants experience confusion and frustration when they try to fix something repeatedly, but it still does not work. This is evidenced by the following excerpt by George.

*"Yeah, I was definitely confused. -- definitely because I did not know what wasn't -- like why it wasn't displaying, and I was trying to look for anything in my code that wasn't working. I don't -- I think I was just like very frustrated, like at the program and like, I was just like looking it over -- and like trying to see like look for mistakes and like, I was like, trying to, um, do different things and like output things -- and like reading over just to see like what the program was doing.."-**George***

George was doing trial and error to fix issues in his code but could not succeed. First, the unexpected code behavior confused him and after multiple unsuccessful attempts, he got frustrated. George's experience explains that students feel confused when they find a discrepancy between their existing mental model of programming knowledge and the actual output [18]. When their confusion is unsolved for a long time, they get frustrated [3, 8].

B. *Triangulation of Multi-Modal Data*

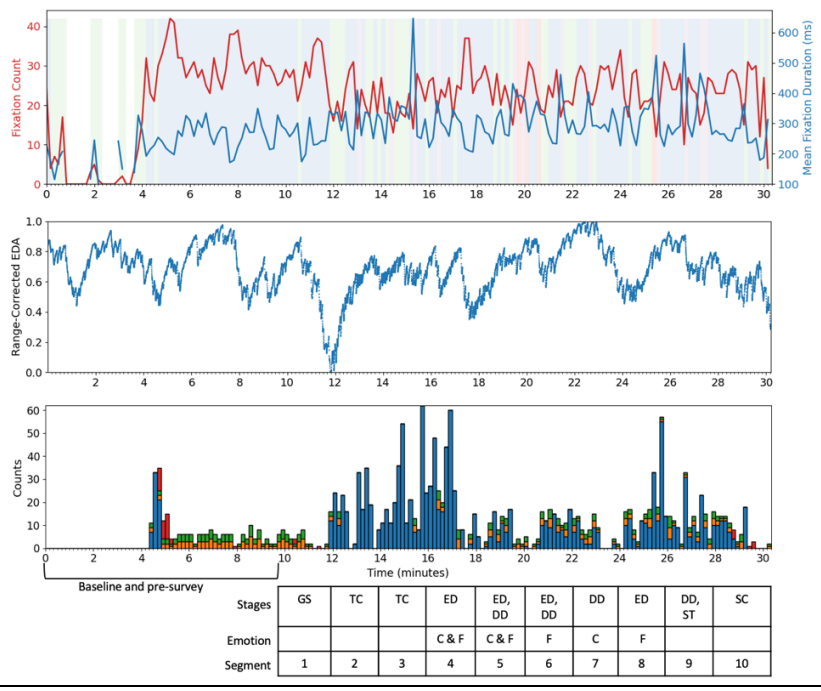
Qualitative findings in the previous section provide an overarching summary of the findings from the retrospective think-aloud interview. However, we selected two exemplars where we provide a more fine-grained, near real-time, and multimodal perspective on how students experience confusion and frustration and what behaviors they exhibit when they experience these emotions. Moreover, these two exemplars are selected based on the successful completion or non-completion of the rainfall problem. These exemplars confirm that for the most part, there is alignment between the different data sources. In Fig. 2 and Fig. 3, in the clickstream and keystroke graph (bottom) blue bars represent the count of keyboard events, while the other colors show mouse events.

Exemplar 1 (Task not completed): Jane is a Hispanic/Latino female student. She was taking CSE-1222 in Fall 2022 for the first time. Before taking this course, she did AP Computer Science. Fig 2 shows Jane's biometric and clickstream graphs and relevant excerpts from the interview data.

Jane's excerpts explain that she started to feel confused during segment 5 [see Fig 2 (b)] when she executed the code and did not get the expected output. In the subsequent segments, she explained that she was trying to figure out the issues in the code which is also evident from Jane's eye-gaze data. As we can see in Jane's eye gaze graph [Fig 2 (a)], during segment 4 and onwards Jane's fixation counts are mostly higher than fixation duration which means that she was looking all around the code screen to find the errors and was not just focusing on particular code sections.

Jane's EDA graphs depict that she was mostly aroused once she encountered difficulties and could not resolve them. However, her EDA started to go down as she was approaching the end of the task. One reason for the change in EDA could be Jane's positive belief that her task was right and that she did her best as explained by her in segment 10 [Fig 2 (b)] [39].

Moreover, Jane's clickstream data also supports the findings from her qualitative data that she was mostly engaged in finding errors rather than typing the code. According to literature, patterns of mouse behavior such as frequency of visiting a page (or section of a page) and duration of visits are highly related to eye gaze behavior and indicate the debugging experience of students [53, 54]. Jane's eye gaze movements for searching errors and frequency of mouse events in the



a

Segment 04: *I was, uh, reading sample run one, and I was just gonna put in the same, um, inputs that, that, um, like the example showed, and then I was going to hopefully get the right, the same average as that one did, which I didn't; [..] I was like, this challenge seemed so easy, how don't -- how didn't I get it right like the first time.*

Segment 05: *I was sitting here like, what am I doing wrong; and I start to get like a little bit worried, like, as in the next like, 10 minutes are gonna go by, and I'm still not gonna figure it out what I did wrong.*

Segment 06: *knowing that I wasn't working properly; I started to question whether the-the samples themselves were wrong, but then I was like, they probably aren't wrong 'cuz they're like, you're probably supposed to use them in order to get the right*

Segment 07: *It was something with the sample numbers that was different than my own numbers, [..] I would figure it out probably 'cuz I was just narrowing down like what was the difference between the sample and my own.*

Segment 08: *And when I got the like divide by zero error, I knew that I could just like, fix that by -- with like an if statement --[..] at first when that came up, I was like, oh great, that's another problem that I have to fix, um, but I just kind of figured I would get it out of the way so I could focus on the bigger problem, which is me not getting the right numbers.*

b

Fig. 3 (a) Jerry's Eye-gaze (top), EDA (middle), and Clickstream and Keystroke (bottom) graphs over the 2-minute segments and the table on the bottom shows the occurrence of confusion (C) and frustration (F) during different stages of the process model (b) Interview excerpts of segments where the participant experienced confusion and frustration.

clickstream graphs [segment 4 and onwards in Fig 2 (a)] show similar patterns thus indicating that she was engaged in debugging, as suggested by literature.

Exemplar 2 (Completed): Jerry is a White male participant. He was taking CSE-1223 in Fall 2022 for the first time. Before taking this course, he had some experience with website development. Fig 3 shows Jerry's biometric and clickstream graphs and relevant excerpts from the interview data. Jerry completed the task successfully but he encountered difficulties during his programming session. During segment 4 in Fig 3 (b) he completed the code and executed it for the first time. His code was executed successfully but he did not get the expected output and experienced confusion and frustration because he perceived that the task was simple and he thought he could do it correctly. His emotional arousal is also evident by peaks in the EDA graph during segment 4 [see Fig 3 (a)]. When Jerry encountered this unexpected behavior, he entered into the debugging process. As we can see in excerpts of segments 5, 6, and 7 in Fig 3 (b) he was trying to resolve the issues. His eye-gaze graph also shows he was searching for issues by looking all over the code [41] as we can see some gaps in fixation counts and duration during segments 5, 6, and 7 [Fig 3 (a)]. Moreover, during these segments, his EDA was fluctuating (not going into baseline but there are changes), which could explain that he was doing trial-and-error, where he might have perceived that he had figured out the issue and updated the code (feeling positive) but could not get the expected output [55]. His clickstream data also shows a mix of mouse and keyboard events during this time. Furthermore, we could confirm from Jerry's EDA data that encountering difficulties in programming could affect physiological data and emotions [55]. For instance, during segment 8 [see Fig 3 (a)] Jerry's EDA data shows an increase, as he got the division by zero error and felt frustration.

V. Discussion

A. Challenges that Trigger Confusion and Frustration

Students reported experiencing confusion and frustration when they encountered challenges like forgetting the syntax of commands (taking user input and initialization of array), not understanding how to do the problem (such as choosing the appropriate loop, sentinel value, and taking multiple user inputs), and not being able to debug logical, or syntax errors. Most of the challenges reflect the problem-solving deficiency of novice students. According to literature, novices face difficulties due to their semantic misconception of programming concepts (incorrect mental models) or their lack of understanding of program composition strategies [1, 46], 52].

As participants write code, they develop some expectations about the output of the code and their progress. When their expectations did not match reality, they feel negative emotions. This was observed in a few participants who could not manage to complete the task in the given time. The literature also suggests that perception about the outcome and negative programming episodes induced negative emotions such as frustration [51]. Moreover, other participants experienced extreme confusion when they encountered difficulties repeatedly. Mainly, two cases were reported: 1) the output of the program did not match the expected output and 2) the participants got errors (syntax errors, division by zero, and infinite loop). Frustration was more prominent when participants forgot the commands such as the import statement and scanner initialization for user input in Java and declaration of the array. To solve the rainfall problem using an array is the least favorable solution because this problem requires storing the sum of inputs rather than each input and also the number of user inputs is unknown [47]. However, using an array is also possible but novice students do not have enough knowledge to use this data structure when the length is unknown [47]. The decision to use an array by participants indicates that novices have inaccurate mental models of programming concepts, which subsequently confused them [1]. As confusion persisted due to the incorrect use of an array, participants became frustrated [3].

Another event that triggered confusion and frustration was when participants did not know how to solve the problem. Participants were able to understand the task requirements and they had a solution in their mind, but they could not translate it into complete code. For instance, Smith (section 2.1) mentioned that he felt frustrated because he was unable to write the code for taking multiple user inputs. Students' difficulty in translating the problem description into code could be explained by the literature that suggests that novice students have knowledge of programming concepts, but they face difficulty in merging different programming concepts to do the programming task [46].

Participants were engaged in dealing with the difficulties stage by doing trial-and-error to find and fix issues in the code. At this stage, participants experienced confusion and frustration when they could not fix issues after repeated trial-and-error. This behavior of students explains that students become confused when they encounter an impasse [3]. This impasse results from the discrepancy between their existing mental models and the actual output [18]. If students make efforts to use their knowledge effectively or get some help to update their mental models, they can resolve confusion and continue to make progress in the code [10]. As in this study, the participants who encountered syntax errors were able to fix them by understanding the error messages and completed the task. However, when the students could not resolve their confusion, they experience frustration and eventually get bored if the frustration persists. For instance, participants in this study experienced frustration when they forgot the commands and eventually got bored because they could not use any help and had nothing to do to make progress in the code [3, 8].

B. Triangulation of Multimodal Data

Triangulation of biometric (eye-gaze and EDA) and behavioral (clickstream and keystroke) data with interview data support our findings of qualitative analysis. This triangulation gives a picture of students' real-time physiological behaviors accompanied by the fine-grained analysis of students' self-report responses, hence enhancing the validity of the findings [13, 34].

From the triangulation, we observed that eye-gaze data could be indicative of students' difficulty in finding useful information or focusing on important sections of code as suggested by [32, 41, 42]. Moreover, some parts of EDA data indicated changes in signals (peaks) when students experienced confusion and frustration. However, changes in EDA data could be due to other emotional experiences such as shame and anger that students reported at the same time. Even so, it confirms the relation between EDA data and emotional arousal [30, 55], and subsequently triangulation of EDA data with students' self-report responses confirmed our

findings. The patterns in clickstream data also validated our findings of the participants' debugging behavior during the programming session. Our data suggests that there are more mouse events when participants try to find issues in code whereas keystroke events are more prominent when participants type code or deal with difficulties along with typing code. Our findings align with previous work [29] that suggests students' log-based data could help in identifying engagement and frustration.

VI. Limitations and Future Directions

This study is conducted in a controlled lab setting which limits the generalizability of findings. However, participants were not required to complete the task to get a grade, or their performance in the task was not associated with any grade which is usually the case when students do programming in their course. Due to this difference in grading mechanism, students may experience certain emotions during course-related programming that they experienced during this study.

Moreover, the institutional review board (IRB) asked to use a secure room and secure system (computer installed with security protocols and without any internet connectivity) to protect participants' privacy. Therefore, it was not possible for the participants to use online help. We also did not provide them with other external help resources (such as lecture notes or tablets) because that would have caused unnecessary movement and hence, would have compromised the quality of the biometric data. Whereas, in real settings, students are not restricted from using resources for help. This difference in the environment may affect the emotional experience of students.

In this study, participants used IDEs that were different from their courses (participants used Zybooks and Repl.it in their courses which were not possible to use on the secure system). Hence, it is possible that some confusion and frustration were also triggered by compilation or IDE-related events.

In this study the challenges for the most frequent emotions of confusion and frustration are explored, however, we observed that students also experience other emotions that impact their performance and perception of their abilities to do the programming. We aim to extend this work by exploring other emotions and their interaction with students' self-efficacy beliefs. Additionally, the triangulation of multimodal data is presented for two participants that we would extend to other participants to find more meaningful patterns of emotional experiences using machine learning techniques.

VII. Implications and Suggestions for Instructors

Understanding students' emotions while programming is important to keep them engaged in solving problems and enhance their learning [16]. In this study, we found that students experience confusion and frustration when they encounter certain problem-solving related challenges such as the use of "for" and "while" loop. To prevent the negative emotions that trigger due to such problem-solving challenges, instructors may need to use appropriate strategies for teaching programming. One strategy is "Problem-based learning", in which teachers could explain worked programming examples that show the different use cases of programming constructs in solving problems [56]. The problem-based learning could facilitate students to build correct mental models of programming constructs [57].

Teachers could also use interactive tools to support the learning of abstract programming concepts. For instance, SICAS2 is a tool that helps students to create programs using flowcharts and see the execution of these programs through the animation [58].

Students find difficulty in converting the problem description into code. One strategy is to teach students "stepwise refinement" which explains how to extract requirements from problem descriptions in the form of goals and subgoals [46]. By identifying the main goal and the subgoals(subproblems) required to solve the problem, students could plan and solve each subgoal individually, making the problem simpler. An important part of this strategy is teaching them how to connect the subgoals. Connecting the subgoals could be facilitated by teaching students the use of flowcharts or explicitly defining the input and output for each subgoal as per the relation among subgoals. However, successful implementation of this strategy relies on students' knowledge and understanding of relevant programming concepts.

VIII. Conclusion

This study provides a fine-grained and near-real-time analysis of how confusion and frustration manifest themselves when students are working on a programming problem. In literature, researchers have mostly discussed the reasons for confusion and frustration while learning to program and how these two emotions transition from each other during learning. In this study, we ask students to do a programming task without providing (teaching) them any learning lesson just before the task. So, students were required to solve the task by using the existing knowledge that they had obtained in their course. We used the rainfall problem because it helps in identifying the basic problem-solving behavior of students. So, our focus was to study the students' problem-solving related challenges that trigger confusion and frustration. Programming instructors could use the findings of this study for an in-depth understanding of what problem-solving techniques students lack so that they could design the appropriate interventions that enhance the conceptual understanding of students.

Acknowledgment

This material is based upon work supported, in part, by the National Science Foundation: NSF IIS 2104729. Any opinions, findings, conclusions, or recommendations expressed in this material do not necessarily reflect those of the NSF.

References

[1] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, Jun. 2003, doi: 10.1076/csed.13.2.137.14200.

[2] I.-H. Hsiao, S. Sosnovsky, and P. Brusilovsky, "Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming: Adaptive navigation support in E-Learning," *Journal of Computer Assisted Learning*, vol. 26, no. 4, pp. 270–283, Jul. 2010, doi: 10.1111/j.1365-2729.2010.00365.x.

[3] N. Bosch and S. D'Mello, "Sequential Patterns of Affective States of Novice Programmers".

[4] R. Pekrun, "The Impact of Emotions on Learning and Achievement: Towards a Theory of Cognitive/Motivational Mediators," *Applied Psychology*, vol. 41, no. 4, pp. 359–376, Oct. 1992, doi: 10.1111/j.1464-0597.1992.tb00712.x.

[5] Z. Atiq and M. Loui, "A Qualitative Study of Emotions Experienced by First-year Engineering Students during Programming Tasks,"

TOCE, vol. 22, no. 3, pp. 1–26, Sep. 2022, doi: https://doi.org/10.1145/3507696.

[6] J. F. Grafsgaard, R. M. Fulton, K. E. Boyer, E. N. Wiebe, and J. C. Lester, "Multimodal analysis of the implicit affective channel in computer-mediated textual communication," in *Proceedings of the 14th ACM international conference on Multimodal interaction*, in ICMI '12. New York, NY, USA: Association for Computing Machinery, Oct. 2012, pp. 145–152. doi: 10.1145/2388676.2388708.

[7] D. M. C. Lee, Ma. M. T. Rodrigo, R. S. J. d. Baker, J. O. Sugay, and A. Coronel, "Exploring the Relationship between Novice Programmer Confusion and Achievement," in *Affective Computing and Intelligent Interaction*, S. D'Mello, A. Graesser, B. Schuller, and J.-C. Martin, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 175–184. doi: 10.1007/978-3-642-24600-5_21.

[8] Z. Liu, V. Pataranutaporn, and J. Ocumpaugh, "Sequences of Frustration and Confusion, and Learning," in *Proceedings of the 6th international conference on educational data mining*, pp. 114–120.

[9] Ma. M. T. Rodrigo *et al.*, "Affective and behavioral predictors of novice programmer achievement," in *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, in ITiCSE '09. New York, NY, USA: Association for Computing Machinery, Jul. 2009, pp. 156–160. doi: 10.1145/1562877.1562929.

[10] S. D'Mello, B. Lehman, R. Pekrun, and A. Graesser, "Confusion can be beneficial for learning," *Learning and Instruction*, vol. 29, pp. 153–170, Feb. 2014, doi: 10.1016/j.learninstruc.2012.05.003.

[11] B. Lehman *et al.*, "Inducing and Tracking Confusion with Contradictions during Complex Learning," *International Journal of Artificial Intelligence in Education*, vol. 22, no. 1–2, pp. 85–105, Jan. 2013, doi: 10.3233/JAI-130025.

[12] S. D'Mello and A. Graesser, "Dynamics of affective states during complex learning," *Learning and Instruction*, vol. 22, no. 2, pp. 145–157, Apr. 2012, doi: 10.1016/j.learninstruc.2011.10.001.

[13] I. V. Alarcón and S. Anwar, "Situating multi-modal approaches in engineering education research," *Journal of Engineering Education*, vol. 111, no. 2, pp. 277–282, 2022, doi: 10.1002/jee.20460.

[14] L. F. Barrett, "The Origin of Feeling," in *How Emotion are Made: The Secret Life of Brain*, Pan Macmillan, 2017.

[15] R. Pekrun, "The Control-Value Theory of Achievement Emotions: Assumptions, Corollaries, and Implications for Educational Research and Practice," *Educ Psychol Rev*, vol. 18, no. 4, pp. 315–341, Nov. 2006, doi: 10.1007/s10648-006-9029-9.

[16] R. Pekrun and E. J. Stephens, "Academic emotions.," in *APA educational psychology handbook, Vol 2: Individual differences and cultural and contextual factors.*, K. R. Harris, S. Graham, T. Urdan, S. Graham, J. M. Royer, and M. Zeidner, Eds., Washington: American Psychological Association, 2012, pp. 3–31. doi: 10.1037/13274-001.

[17] N. Li, J. D. Kelleher, and R. Ross, "Detecting Interlocutor Confusion in Situated Human-Avatar Dialogue: A Pilot Study," 2021, doi: 10.21427/bsd0-7326.

[18] S. K. D'Mello and A. C. Graesser, "Confusion," in *International Handbook of Emotions in Education*, Routledge, 2013. doi: 10.4324/9780203148211.ch15.

[19] S. D'Mello and A. Graesser, "Confusion and its dynamics during device comprehension with breakdown scenarios," *Acta Psychologica*, vol. 151, pp. 106–116, Sep. 2014, doi: 10.1016/j.actpsy.2014.06.005.

[20] R. A. Calvo and S. D'Mello, "Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications," *IEEE Transactions on Affective Computing*, vol. 1, no. 1, pp. 18–37, Jan. 2010, doi: 10.1109/T-AFFC.2010.1.

[21] G. V. Caprara *et al.*, "Longitudinal analysis of the role of perceived self-efficacy for self-regulated learning in academic continuance and achievement.," *Journal of Educational Psychology*, vol. 100, no. 3, pp. 525–534, Aug. 2008, doi: 10.1037/0022-0663.100.3.525.

[22] N. Bosch and S. D'Mello, "The Affective Experience of Novice Computer Programmers," *Int J Artif Intell Educ*, vol. 27, no. 1, pp. 181–206, Mar. 2017, doi: 10.1007/s40593-015-0069-5.

[23] R. Pekrun, "Self-Report is Indispensable to Assess Students' Learning," *FLR*, vol. 8, no. 3, pp. 185–193, Mar. 2020, doi: 10.14786/flr.v8i3.637.

[24] Ma. M. Rodrigo, J. Sugay, R. Baker, and E. Tabanao, "Monitoring Novice Programmer Affect and Behaviors to Identify Learning Bottlenecks," *Department of Information Systems & Computer Science Faculty Publications*, Jan. 2009, [Online]. Available: https://archium.ateneo.edu/discs-faculty-pubs/123

[25] "It's Written on Your Face: Detecting Affective States from Facial Expressions while Learning Computer Programming | SpringerLink." https://link.springer.com/chapter/10.1007/978-3-319-07221-0_5 (accessed Apr. 21, 2023).

[26] J. F. Grafsgaard, K. E. Boyer, and J. C. Lester, "Predicting Facial Indicators of Confusion with Hidden Markov Models," in *Affective Computing and Intelligent Interaction*, S. D'Mello, A. Graesser, B. Schuller, and J.-C. Martin, Eds., in Lecture Notes in Computer Science, vol. 6974. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 97–106. doi: 10.1007/978-3-642-24600-5_13.

[27] J. F. Grafsgaard, J. B. Wiggins, K. E. Boyer, E. N. Wiebe, and J. C. Lester, "Automatically Recognizing Facial Indicators of Frustration: A Learning-centric Analysis," in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, Sep. 2013, pp. 159–165. doi: 10.1109/ACII.2013.33.

[28] T. J. Tiam-Lee and K. Sumi, "Adaptive Feedback Based on Student Emotion in a System for Programming Practice," in *Intelligent Tutoring Systems*, R. Nkambou, R. Azevedo, and J. Vassileva, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 243–255. doi: 10.1007/978-3-319-91464-0_24.

[29] T. J. Tiam-Lee and K. Sumi, "Analysis and Prediction of Student Emotions While Doing Programming Exercises," in *Intelligent Tutoring Systems*, A. Coy, Y. Hayashi, and M. Chang, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 24–33. doi: 10.1007/978-3-030-22244-4_4.

[30] A. Haag, S. Goronzy, P. Schaich, and J. Williams, "Emotion Recognition Using Bio-sensors: First Steps towards an Automatic System," in *Affective Dialogue Systems*, E. André, L. Dybkjær, W. Minker, and P. Heisterkamp, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 36–48. doi: 10.1007/978-3-540-24842-2_4.

[31] S. T. Iqbal, X. S. Zheng, and B. P. Bailey, "Task-evoked pupillary response to mental workload in human-computer interaction," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, in CHI EA '04. New York, NY, USA: Association for Computing Machinery, Apr. 2004, pp. 1477–1480. doi: 10.1145/985921.986094.

[32] Z. Sharafi, B. Sharif, Y.-G. Guéhéneuc, A. Begel, R. Bednarik, and M. Crosby, "A practical guide on conducting eye tracking studies in software engineering," *Empir Software Eng*, vol. 25, no. 5, pp. 3128–3174, Sep. 2020, doi: 10.1007/s10664-020-09829-4.

[33] M. Pachman, A. Arguel, L. Lockyer, G. Kennedy, and J. Lodge, "Eye tracking and early detection of confusion in digital learning environments: Proof of concept," *AJET*, vol. 32, no. 6, Dec. 2016, doi: 10.14742/ajet.3060.

[34] I. V. Alarcón, S. Anwar, and Z. Atiq, "How Multi-modal Approaches Support Engineering and Computing Education research," unpublished.

[35] W. Sun and X. Sun, "Teaching Computer Programming Skills to Engineering and Technology Students with a Modular Programming Strategy," in *2011 ASEE Annual Conference & Exposition Proceedings*, Vancouver, BC: ASEE Conferences, Jun. 2011, p. 22.1378.1-22.1378.11. doi: 10.18260/1-2--18625.

[36] M. Q. Patton, *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*. SAGE Publications, 2014.

[37] S. Elling, L. Lentz, and M. de Jong, "Retrospective think-aloud method: using eye movements as an extra cue for participants' verbalizations," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, in CHI '11. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 1161–1170. doi: 10.1145/1978942.1979116.

[38] A. Hyrskykari, S. Ovaska, P. Majaranta, K.-J. Räihä, and M. Lehtinen, "Gaze Path Stimulation in Retrospective Think-Aloud," *JEMR*, vol. 2, no. 4, Nov. 2008, doi: 10.16910/jemr.2.4.5.

[39] J. M. Basch and C. Fisher, "Affective events - emotions matrix: a classification of work events and associated emotions," 1998. Accessed: Mar. 03, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Affective-events-emotions-matrix%3A-a-classification-Basch-Fisher/660d4211ba7b0b80a9370b2b01aba7df482922ce

[40] T. Fritz and S. C. Muller, "Leveraging Biometric Data to Boost Software Developer Productivity," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*

(SANER), Suita, Osaka, Japan: IEEE, Mar. 2016, pp. 66–77. doi: 10.1109/SANER.2016.107.

[41] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "A Survey on the Usage of Eye-Tracking in Computer Programming," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–58, Jan. 2019, doi: 10.1145/3145904.

[42] F. Hauser, J. Mottok, and H. Gruber, "Eye Tracking Metrics in Software Engineering," in *Proceedings of the 3rd European Conference of Software Engineering Education*, in ECSEE'18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 39–44. doi: 10.1145/3209087.3209092.

[43] D. Girardi, F. Lanubile, and N. Novielli, "Emotion detection using noninvasive low cost sensors," in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, San Antonio, TX: IEEE, Oct. 2017, pp. 125–130. doi: 10.1109/ACII.2017.8273589.

[44] R. Hanamanthrao and S. Thejaswini, "Real-time clickstream data analytics and visualization," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, May 2017, pp. 2139–2144. doi: 10.1109/RTEICT.2017.8256978.

[45] "iMotions: Unpack Human Behavior," *Imotions*. https://imotions.com/ (accessed Feb. 09, 2022).

[46] E. Soloway, "Learning to program = learning to construct mechanisms and explanations," *Communications of the ACM*, vol. 29, no. 9, pp. 850–858, Sep. 1986, doi: 10.1145/6592.6594.

[47] K. Fisler, "The recurring rainfall problem," in *Proceedings of the tenth annual conference on International computing education research - ICER '14*, Glasgow, Scotland, United Kingdom: ACM Press, 2014, pp. 35–42. doi: 10.1145/2632320.2632346.

[48] P. Kather, R. Duran, and J. Vahrenhold, "Through (Tracking) Their Eyes: Abstraction and Complexity in Program Comprehension," *ACM Trans. Comput. Educ.*, vol. 22, no. 2, pp. 1–33, Jun. 2022, doi: 10.1145/3480171.

[49] R. Pekrun and L. Linnenbrink-Garcia, *International Handbook of Emotions in Education*. 2014.

[50] H.-F. Hsieh and S. E. Shannon, "Three Approaches to Qualitative Content Analysis," *Qual Health Res*, vol. 15, no. 9, pp. 1277–1288, Nov. 2005, doi: 10.1177/1049732305276687.

[51] P. Kinnunen and B. Simon, "CS majors' self-efficacy perceptions in CS1: results in light of social cognitive theory," in *Proceedings of the seventh international workshop on Computing education research*, in ICER '11. New York, NY, USA: Association for Computing Machinery, Aug. 2011, pp. 19–26. doi: 10.1145/2016911.2016917.

[52] A. Ebrahimi, "Novice programmer errors: language constructs and plan composition," *International Journal of Human-Computer Studies*, vol. 41, no. 4, pp. 457–480, Oct. 1994, doi: 10.1006/ijhc.1994.1069.

[53] "Eye Gaze and Mouse Cursor Relationship in a Debugging Task | SpringerLink." https://link.springer.com/chapter/10.1007/978-3-642-39473-7_93 (accessed May 15, 2023).

[54] "Eye-mouse coordination patterns on web search results pages | CHI '08 Extended Abstracts on Human Factors in Computing Systems." 10.1145/1358628.135879 (accessed May 15, 2023).

[55] J. Gorson, K. Cunningham, M. Worsley, and E. O'Rourke, "Using Electrodermal Activity Measurements to Understand Student Emotions While Programming," in *Proceedings of the 2022 ACM Conference on International Computing Education Research V.1*, Lugano and Virtual Event Switzerland: ACM, Aug. 2022, pp. 105–119. doi: 10.1145/3501385.3543981.

[56] I. de Jong and J. Jeuring, "Computational Thinking Interventions in Higher Education: A Scoping Literature Review of Interventions Used to Teach Computational Thinking," in *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, in Koli Calling '20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 1–10. doi: 10.1145/3428029.3428055.

[57] A. West and P. Lombardi, "Scaffolding," in *Instructional Methods, Strategies and Technologies to Meet the Needs of All Learners*,

[58] A. Ferreira, A. Gomes, and A. J. Mendes, "SICAS2: Interactive Tool to Support Programming Learning," in *2022 International Symposium on Computers in Education (SIIE)*, Nov. 2022, pp. 1–5. doi: 10.1109/SIIE56031.2022.9982323.