ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #37268

Work in Progress: Understanding CS1 Students' Code Comprehension Behaviors using Multi-modal Data

FNU Rakhi

NA

Syedah Zahra Atiq (Assistant Professor of Practice) (The Ohio State University)

I am an Assistant Professor of Practice at the Ohio State University

© American Society for Engineering Education, 2022
Powered by www.slayte.com

Understanding CS1 Students' Code Comprehension Behaviors using Multi-modal Data

1. Introduction

Code comprehension is an important skill for programmers because it helps them understand code and develop debugging skills [1]. The process of code comprehension is unlike comprehending natural languages because it involves complex cognitive processing. During cognitive processing, a programmer is required to develop or use the appropriate mental models of programming constructs, which makes code comprehension difficult for novice programmers [2]. Along with cognitive processing, it is important to analyze how students feel during code comprehension because the literature suggests that emotions influence different aspects of cognition such as attention, reasoning, learning, memory, and problem-solving [3]. Novice programmers may experience a variety of emotions while comprehending code. These changes in emotions may subsequently influence their academic performance and retention in computing and engineering [4]. Therefore, in this study, we aim to understand CS1 students' emotions and cognitive processing during code comprehension. Specifically, we ask the following research questions:

1. What type of cognitive processing do CS1 students perform during code comprehension?
2. What emotions do CS1 students experience during code comprehension?
3. How do programmers' emotions and cognitive processing interact during code comprehension?

Answers to these research questions would provide us with an in-depth and nuanced understanding of the cognitive events that trigger certain emotions and how students process that information, and vice versa.

In this study, we will employ multi-modal data, collected through biometric sensors and concurrent think-aloud interviews. These data would provide multiple perspectives and a rich understanding of the instructional needs of CS1 students by analyzing their emotions and cognitive processing during code comprehension. These instructional needs may include demonstrating programming concepts with examples, construction of mental models through visualization, debugging strategies, and scaffolding [5]. Based on the instructional needs, appropriate instructional strategies (pedagogical, technological, or content-based) could be designed that may provide students with good learner support [5].

2. Literature Review

2.1. Theoretical Framework

This research study is grounded in two theoretical frameworks: Cognitive load theory (CLT) and Control-value theory of achievement emotions (CVT). In the following sections, we briefly explain these theories.

2.1.1. Cognitive Load Theory: According to CLT, cognitive load is a construct that measures the load, imposed by a task on the cognitive system of an individual [6]. The load can be characterized as task-based (mental load) and learner-based (mental effort). Task-based cognitive load is imposed by task demands. These demands could be due to the complexity of the task or the

instructional design [6]. Learner-based cognitive load indicates the amount of cognitive capacity allocated by the person to meet the task demands [6].

Learning to program is a cognitively demanding task because it imposes cognitive requirements such as mental load and mental effort [5]. Solving a programming task requires an understanding of the program, its syntax, and semantics, understanding the developing environment, checking code output, compilation, and debugging. The simultaneous holding of this information in working memory imposes the cognitive load and makes the programming task demanding [5].

2.1.2. Control Value Theory of Achievement Emotions: Control value theory (CVT) provides a theoretical framework to study emotions in the academic context. It suggests that academic emotions influence academic performance and learning, and they link directly with achievement activities and outcomes [4]. According to CVT, achievement emotions are studied in three dimensions i.e., object focus, activation, and valence. Object focus refers to the activity students engage in or the outcome of the activity. Valance dimension describes emotion as pleasant/positive or unpleasant/negative whereas activation refers to the arousal of emotion due to a physiological response to a particular emotion. These dimensions categorize the emotions in four categories (pleasant/activating, unpleasant/activating pleasant/deactivating, and unpleasant/deactivating) [4].

2.2. Reasons why Repetition Structures are Hard: There are various reasons that make the repetition structures hard to learn. In this section, we define three main reasons. *The first reason is that novice programmers do not have prior mental models to learn programming constructs* [7]. The absence of mental models leads to misconceptions caused by inappropriate memory transfer [8]. These misconceptions occur when a certain term in programming does not have the same meaning as it has in the English language. Various programming languages use words like "while" and "for", to represent loops, which may not have the same meaning for programming as they have in the English language or may have multiple meanings in English [8]. For instance, "while" in the English language can be used as a noun, conjunction, adverb, and verb. However, in programming, it means that the computer will repeat a statement or set of statements while a certain condition remains true. This may confuse a novice, who will initially try to use their existing mental models from English to understand "while" and "for" in programming.

The second reason is dealing with troublesome cases and skipping certain values. A simple loop that displays a list of numbers on the screen may not be very hard for the student to grasp. However, the literature suggests that in some cases, loops become extremely difficult for novices, and sometimes even for expert programmers. A classic scenario in which loops are difficult for students to learn is the rainfall problem [9, 10]. This problem uses the while loop to read the integers(rainfall) as input and output the average of these integers. While computing the average, the program excludes negative numbers and stops when the input is 99999. These types of loops are hard because students must accurately convert the problem into code while taking into consideration special cases, which may not be apparent to them initially. This is more a challenge of proper problem understanding than a matter of correct syntax.

The third reason is the manipulation of the control variable of the loop, especially in the "for" loop. In the "for" loop, the value of the control variable changes on each iteration of the loop. This change in value could be sequential (increment or decrement by a constant number) or may involve

an expression. Since the value of the control variable is hidden, the programmer may fail to see the internal changes to the value [11, 12].

2.3. Emotions and Cognition: Emotional experiences play an important role in an academic setting because they modulate different aspects of cognition such as attention, reasoning, learning, memory, and problem-solving [13-17]. During exams, tests, and projects, students engage in cognitive processing tasks, and these activities are associated with emotional states of anxiety, frustration, and boredom [3]. Moreover, the subject of study also influences emotions that affect the ability of a person to learn and remember. For instance, in introductory programming courses, students face many difficulties like understanding programming concepts, the syntax of the language, and debugging [7]. Students may feel many different emotions because of these difficulties, subsequently influencing their learning and academic performance [4]. Therefore, it is important to design courses by considering the influence of emotions and maximizing the learning and retention of subject knowledge [18].

2.4. Code Comprehension: Programming is not a single task, it involves multiple processes such as reading, comprehending, tracing, summarizing, writing, and debugging the code [19]. In this study, we will focus on one such aspect of programming, which is code comprehension. Comprehension is usually described as a process in which an individual constructs his or her mental representation of the program [20]. During code comprehension, students deal with many concepts and integrate them to form a mental model of the dynamic aspects of the program execution [21].

2.4.1. Data Collection Methods for Code Comprehension: Research studies in programming for code comprehension have used different methods for data collection such as questionnaires, interview-based methods, and eye-tracking [22, 23]. In a study by the Leeds group, the authors created a questionnaire of code comprehension tasks to investigate why students find programming hard [24]. They found that many students performed poorly in code comprehension problems, "suggesting that such students have a fragile grasp of skills that are a pre-requisite for problem-solving". Later, Whalley, J.L et al., claimed that the choice of code comprehension tasks by the Leeds group was not informed by the theoretical model [25]. Therefore, they extended the work of the Leeds group by developing a set of code comprehension tasks based on SOLO and Bloom taxonomies. From the analysis of their framework, they found a relation between the cognitive level of questions with the performance of students.

Adelson prepared a set of eight code comprehension questions in pascal to compare the performance of novice and expert computing students [26]. The author found that experts always outperformed on well-written tasks while on unstructured (poorly written) tasks, sometimes novices outperformed experts. Cynthia and Susan created six short programming segments to identify what kind of information novice students used to comprehend the program and how they connected different parts of information [27]. Their findings suggest that novice students create concrete and detailed mental representations of code during comprehension. In recent years, eye-tracking technology has gained attention as a method for data collection for code comprehension studies [23]. The rationale is that eye-tracking provides near real-time and in-depth insights into the cognitive processes that the programmer engages in while comprehending the code [1].

2.4.1.1. Eye Movement: Eye movement helps to understand the person's cognitive processing by observing the viewing pattern of a person on the stimulus [23]. The most important and frequently used eye movement metrics are fixation duration, fixation count, saccades, and scan path [28]. *Fixation* tells us where the person is looking at a particular time [23]. Fixation can last for milliseconds or up to several seconds, the total time of fixation is called *fixation duration*. Fixation duration is associated with the attention being paid to the stimulus. The longer the duration, the more interesting or complex is the stimulus [28]. A *saccade* is a rapid movement of the eye between fixations, it helps to understand the viewing pattern of the person on the screen [23]. A *scan path* is a combination of saccades and fixations. It is described as a path formed by the directed sequence of saccades between fixations [28]. The eye movement metrics are usually measured for part(s) of the screen (such as a section of code) where the researcher wants to observe a person's behavior. The selected part of the screen is known as the *area of interest (AOI)* [23]. Another metric that is measured through eye trackers is *pupil dilation* which refers to the change in pupil size due to the autonomic nervous system (ANS) activity [29]. Change in pupil size is associated with emotional arousal in the human body [30-32].

2.4.1.2. Understanding Cognitive Processing using Eye Movements: A range of studies have used eye-movement metrics (fixation duration, fixation count, saccades, and scan path) to analyze the cognitive processing [23]. Christoph and Crosby used eye-tracking to study the code comprehension patterns of novice programmers and expert programmers [33]. They used eye fixation as a measure of attention and found that highly experienced developers use less time on comments and more time on complex statements compared to novice developers.

Just and Carpenter identified the relation between fixation of the eye and cognitive processing of students during comprehension of scientific passages [34]. They measured the level of cognitive processing of a text by calculating fixation duration on each word of text [34]. Chen et al., used eye blinks, fixation, and saccades to measure the cognitive load. They found that eye blinks got stuck during attention-demanding tasks to maximize the stimulus perception. Whereas fixation and saccades metrics were controlled by the effort required to spread attention on the task-related objects [35]. Recently, Philipp et al., used fixation time on areas of interest (AOI) to identify the effect of code composition style and familiarity of code on the complexity of code comprehension [21].

2.5. Understanding Emotions through Biometric Data: Besides eye-tracking for identifying the cognitive processing of programmers, biometric data such as electroencephalography (EEG), electrodermal activity (EDA), blood volume pulse (BVP), and heart rate (HR) could be used to analyze the programmers' emotions [30-32]. Biometric data represent the activity of the human body's automatic nervous system and central nervous system. Some forms of these data may represent changes in emotions along with other nervous system activities [36, 37]. For instance, signals from electrodermal activity (EDA) represent the changes in levels of sweat produced by the human body. EDA signal has two components i.e., tonic, and phasic. Tonic signals show the basic level of skin conductance in the human body whereas phasic signals show the changes in skin conductance due to external stimuli like noises, sound, and lighting [38]. The phasic signal also reflects changes in emotional intensity like high arousal [38].

In literature, a few studies have used biometric data to analyze the programmers' emotions. Girardi et al., used EEG, EDA, BVP, and HR to identify emotions experienced by the developers and the events that trigger those emotions [39]. They found that although developers experience a wide range of emotions during programming tasks, emotions related to negative valence and high arousal were most common. These emotions were triggered by unexpected code behavior, missing documentation, time pressure, and self-perceived low productivity. Fritz and Muller conducted multiple studies to analyze the perceived progress of a developer, task difficulty, and boosting the developer's productivity by using biometric data [30-32]. They found that developers experience many emotions which affect their perceived progress. Further, they suggested that identifying the emotions of developers may help to improve their productivity. For instance, by providing suggestions and resources when developers are found stuck and make no progress.

2.6. Gaps in Literature: From the literature review, it is evident that eye-tracking metrics are useful to understand cognitive processing but measuring cognitive load is not a simple task. It is a multidimensional construct and needs multiple sources to confirm the accuracy of the measure [6]. Therefore, in this study, we will use eye movement metrics along with concurrent think-aloud interviews with participants to understand the cognitive processing of students as they comprehend code. Along with cognitive processing, it is important to analyze how students feel during code comprehension because the literature suggests that emotions influence different aspects of cognition such as attention, reasoning, learning, memory, and problem-solving. Therefore, in this study, we aim to understand CS1 students' cognitive processing as well as emotions during code comprehension by using multi-modal data to get multiple perspectives and a rich understanding of emotions and cognitive processing that students experience while comprehending code.

3. Research Methodology

3.1. Context and Participants: The Ohio State University (OSU) is a large university in the midwestern United States. It is well-known for engineering and has traditional-aged undergraduate students. The Computer Science and Engineering department offers a suite of introductory programming courses in different languages (e.g., C++, Python, and Java). The context for this study is the introductory programming course in Java (CSE-1223). CSE-1223 is offered year-round at OSU and the student population in each semester is different. This course is offered to students from all majors including students who wish to pursue CS as a major. CSE-1223 is taught using active, blended, and project-based learning methodologies.

3.2. Selection Criteria: For this study, novice programming students who would be taking CSE-1223 for the first time during the summer or autumn 2022 semesters would be considered. In the context of this study, novices are students who have not had any programming experience before taking CSE-1223, and they may or may not be taking another programming course in parallel. Since this is a small-scale exploratory study, we will perform purposive sampling from the selected novice students. Through sampling, we will recruit 20 students that ensure diversity and multiple perspectives [40]. Students' emotions may be influenced by the background factors, which as result may impact their academic performance and retention in computing [41]. Therefore, we will consider diversity regarding gender, major field of study, and ethnicity during sampling.

3.3. Research Design: For this study, we will present two short programs written in java to the participants. These programs will be based on loops because loops are hard to understand for various reasons. Some of the reasons include handling the loop variable, dealing with troublesome cases, skipping certain values, and different contextual meaning of loop keywords "for" and "while" in the English language. The primary task for the participants is to comprehend these programs. While comprehending code, participants will concurrently talk about their thought process, which will be audio recorded. Additionally, we will collect participants' biometric data (eye movements, pupil dilation, and EDA). Figure 1 describes the research design.

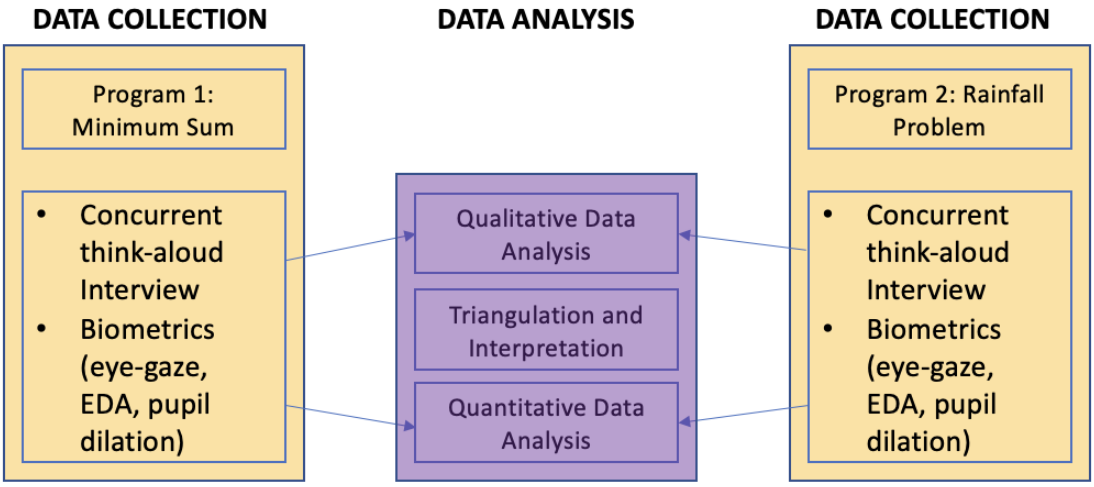

Figure 1: Research Design of Study

3.3.1. Programming Tasks: The two short programs that we are planning to use in this study are the minimum sun and the rainfall problem. The *minimum sum problem* provides the program with an array of integers as input and adds the sum of all elements of the array and the minimum value of the array. The *rainfall program* uses a sentinel-controlled while loop to read the integers (rainfall) as input and output the average of these integers. The sentinel condition stops taking input when the user enters 99999. Additionally, the program excludes negative numbers while calculating the average. These programs are appropriate for this study as they test students' conceptual understanding of loops and other basic programming concepts such as reading input from users, conditional statements, and operators [21, 42, 43]. Furthermore, these programs have been extensively used in research studies related to code comprehension and hence these problems have been tested for their complexity and appropriateness [21].

3.3.2. Data Collection Methods: To answer our research questions, we will collect multiple forms of data. First, we will audio record the participants' verbal think-aloud responses as they comprehend parts of the code. This technique is called a concurrent think-aloud interview [44]. Concurrent think-aloud interview enables students to think aloud about their cognitive and emotional processes as they are in the process of comprehension instead of after the task to avoid the loss of information [44]. Second, we will collect multiple forms of biometric data (eye-gaze, EDA, and pupil dilation). These data provide near real-time information about participants' cognitive functioning, and emotional arousal. For biometric data, we will use iMotions [45] with

non-invasive devices such as Shimmer [46] to capture electrodermal activity (EDA) for emotion arousal, and Smart Eye eye-tracker [47] for cognitive processes.

3.3.3. Data Analysis Methods: For the data analysis of this study, we plan to perform both qualitative and quantitative analysis.

3.3.3.1. Quantitative Data Analysis: Biometric data will be used to perform quantitative data analysis. We will identify areas of interest (AOI) on the tasks and calculate the fixation duration and saccades on those areas from the eye tracker. These metrics will help to understand the students' cognitive processing. For emotion analysis, we will use EDA signals and pupil dilations as quantitative measurements because previous research studies have identified them as a measure of emotional arousal [30-32].

3.3.3.2. Qualitative Data Analysis: We will use thematic analysis to analyze data from the concurrent-think-aloud interview [44]. The goal of qualitative data analysis is to supplement the findings of biometric data and provide more insights into students' emotions and cognitive processes during programming.

3.3.3.3. Triangulation of Quantitative and Qualitative Findings: Once we have conducted both qualitative and quantitative data analysis, we will triangulate the findings of both types of analyses. Our team has developed a qualitative protocol to triangulate biometric data with qualitative excerpts [48]. This protocol temporally aligns graphic visualizations of biometrics with qualitative excerpts and then analyzes data simultaneously to assess for convergence and divergence. Since the context in this study has changed, the triangulation protocol will be adapted to help answer the current research questions.

4. Future Direction

The immediate future directions for this study include data collection during the summer and fall of 2022 and data analysis during spring 2023.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant 2104729. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. Busjahn, C. Schulte, and A. Busjahn, "Analysis of code reading to gain more insight in program comprehension," in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research - Koli Calling '11*, Koli, Finland, 2011, p. 1. DOI: 10.1145/2094131.2094133.

[2] T. Busjahn *et al.*, "Eye Movements in Code Reading: Relaxing the Linear Order," in *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, Italy, 2015, pp. 255–265. DOI: 10.5555/2820282.2820320.

[3] C. M. Tyng, H. U. Amin, M. N. M. Saad, and A. S. Malik, "The Influences of Emotion on Learning and Memory," *Front. Psychol.*, vol. 8, p. 1454, Aug. 2017, DOI: 10.3389/fpsyg.2017.01454.

[4] R. Pekrun and R. P. Perry, "Control-Value Theory of Achievement Emotions," in *International Handbook of Emotions in Education*, 1st ed., 2014, pp. 130–151.

[5] I. T. C. Mow, "Issues and Difficulties in Teaching Novice Computer Programming," in *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*, M. Iskander, Ed. Dordrecht: Springer Netherlands, 2008, pp. 199–204. DOI: 10.1007/978-1-4020-8739-4_36.
[6] J. Sweller, J. J. G. van Merrienboer, and F. G. W. C. Paas, "Cognitive Architecture and Instructional Design," *Educ. Psychol. Rev.*, vol. 10, no. 3, pp. 251–296, Sep. 1998, DOI: 10.1023/A:1022193728205.
[7] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, Jun. 2003, DOI: 10.1076/csed.13.2.137.14200.
[8] Michael Clancy, "Misconceptions and attitudes that interfere with learning to program.," in *Computer science education research*, Taylor & Francis., 2005, pp. 95–110.
[9] M. Guzdial, "Learner-Centered Design of Computing Education: Research on Computing for Everyone," *Synth. Lect. Hum.-Centered Inform.*, vol. 8, no. 6, pp. 1–165, Nov. 2015, DOI: 10.2200/S00684ED1V01Y201511HCI033.
[10] E. Soloway, J. Bonar, and K. Ehrlich, "Cognitive strategies and looping constructs: an empirical study," *Commun. ACM*, vol. 26, no. 11, pp. 853–860, Nov. 1983, DOI: 10.1145/182.358436.
[11] B. Du Boulay, "Some Difficulties of Learning to Program," *J. Educ. Comput. Res.*, vol. 2, no. 1, pp. 57–73, Feb. 1986, doi: 10.2190/3LFX-9RRF-67T8-UVK9.
[12] T. Sirkiä and J. Sorva, "Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises," in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research - Koli Calling '12*, Koli, Finland, 2012, pp. 19–28. DOI: 10.1145/2401796.2401799.
[13] P. Vuilleumier, "How brains beware: neural mechanisms of emotional attention," *Trends Cogn. Sci.*, vol. 9, no. 12, pp. 585–594, Dec. 2005, DOI: 10.1016/j.tics.2005.10.011.
[14] E. A. Phelps, "Human emotion and memory: interactions of the amygdala and hippocampal complex," *Curr. Opin. Neurobiol.*, vol. 14, no. 2, pp. 198–202, Apr. 2004, DOI: 10.1016/j.conb.2004.03.015.
[15] Um, E., Plass, J.L, Hayward, E.O, and Homer, B.D, "Emotional design in multimedia learning," *J. Educ. Psychol.*, vol. 104, no. 2, p. 485, 2012, DOI: https://doi.org/10.1037/a0026609.
[16] N. Jung, C. Wranke, K. Hamburger, and M. Knauff, "How emotions affect logical reasoning: evidence from experiments with mood-manipulated participants, spider phobics, and people with exam anxiety," *Front. Psychol.*, vol. 5, 2014, DOI: 10.3389/fpsyg.2014.00570.
[17] Isen, A. M, Daubman, K. A, and Nowicki, G. P., "Positive affect facilitates creative problem solving.," *J. Pers. Soc. Psychol.*, vol. 52, no. 6, pp. 1122–1131, 1987, DOI: 10.1037%2F0022-3514.52.6.1122.
[18] L. Shen, M. Wang, and R. Shen, "Affective e-Learning: Using 'Emotional' Data to Improve Learning in Pervasive Learning Environment," *J. Educ. Technol. Soc.*, vol. 12, no. 2, pp. 176–189, 2009.
[19] C. Izu *et al.*, "Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories," in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, Aberdeen Scotland Uk, Dec. 2019, pp. 27–52. DOI: 10.1145/3344429.3372501.
[20] C. Schulte, T. Clear, A. Taherkhani, T. Busjahn, and J. H. Paterson, "An introduction to program comprehension for computer science educators," in *Proceedings of the 2010 ITiCSE working group reports*, New York, NY, USA, Jun. 2010, pp. 65–86. DOI: 10.1145/1971681.1971687.
[21] 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.
[22] J. Sheard, S. Simon, M. Hamilton, and J. Lönnberg, "Analysis of research into the teaching and learning of programming," in *Proceedings of the fifth international workshop on Computing education research workshop - ICER '09*, Berkeley, CA, USA, 2009, p. 93. DOI: 10.1145/1584322.1584334.
[23] 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.
[24] R. Lister *et al.*, "A multi-national study of reading and tracing skills in novice programmers," *ACM SIGCSE Bull.*, vol. 36, no. 4, pp. 119–150, Jun. 2004, DOI: 10.1145/1041624.1041673.
[25] J. L. Whalley *et al.*, "An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies," Dec. 2006, Accessed: May 12, 2022. [Online]. Available: https://opus.lib.uts.edu.au/handle/10453/5050
[26] B. Adelson, "When novices surpass experts: The difficulty of a task may increase with expertise.," *J. Exp. Psychol. Learn. Mem. Cogn.*, vol. 10, no. 3, p. 483, 19850101, DOI: 10.1037/0278-7393.10.3.483.
[27] C. L. Corritore and S. Wiedenbeck, "What do novices learn during program comprehension?" *Int. J. Human–Computer Interact.*, vol. 3, no. 2, pp. 199–222, Jan. 1991, DOI: 10.1080/10447319109526004.

[28] F. Hauser, J. Mottok, and H. Gruber, "Eye Tracking Metrics in Software Engineering," in *Proceedings of the 3rd European Conference of Software Engineering Education*, New York, NY, USA, Jun. 2018, pp. 39–44. DOI: 10.1145/3209087.3209092.
[29] P. Ren, A. Barreto, J. Huang, Y. Gao, F. R. Ortega, and M. Adjouadi, "Off-line and On-line Stress Detection Through Processing of the Pupil Diameter Signal," *Ann. Biomed. Eng.*, vol. 42, no. 1, pp. 162–176, Jan. 2014, DOI: 10.1007/s10439-013-0880-9.
[30] 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, Mar. 2016, pp. 66–77. DOI: 10.1109/SANER.2016.107.
[31] S. C. Muller and T. Fritz, "Stuck and Frustrated or in Flow and Happy: Sensing Developers' Emotions and Progress," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy, May 2015, pp. 688–699. DOI: 10.1109/ICSE.2015.334.
[32] S. C. Müller and T. Fritz, "Using (bio)metrics to predict code quality online," in *Proceedings of the 38th International Conference on Software Engineering*, Austin Texas, May 2016, pp. 452–463. DOI: 10.1145/2884781.2884803.
[33] A, Christoph, and M Crosby, "Code scanning patterns in program comprehension," presented at the Proceedings of the 39th Hawaii international conference on system sciences, 2006.
[34] M. A. Just and P. A. Carpenter, "A theory of reading: From eye fixations to comprehension.," *Psychol. Rev.*, vol. 87, no. 4, pp. 329–354, 1980, DOI: 10.1037/0033-295X.87.4.329.
[35] S. Chen, J. Epps, N. Ruiz, and F. Chen, "Eye activity as a measure of human mental effort in HCI," in *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '11*, Palo Alto, CA, USA, 2011, p. 315. DOI: 10.1145/1943403.1943454.
[36] W. Szwoch, "Emotion Recognition Using Physiological Signals," in *Proceedings of the Multimedia, Interaction, Design and Innovation*, New York, NY, USA, Jun. 2015, pp. 1–8. DOI: 10.1145/2814464.2814479.
[37] L. Li and J. Chen, "Emotion Recognition Using Physiological Signals," in *Advances in Artificial Reality and Tele-Existence*, vol. 4282, Z. Pan, A. Cheok, M. Haller, R. W. H. Lau, H. Saito, and R. Liang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 437–446. DOI: 10.1007/11941354_44.
[38] 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, Oct. 2017, pp. 125–130. DOI: 10.1109/ACII.2017.8273589.
[39] D. Girardi, N. Novielli, D. Fucci, and F. Lanubile, "Recognizing developers' emotions while programming," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, New York, NY, USA, Jun. 2020, pp. 666–677. DOI: 10.1145/3377811.3380374.
[40] J. Walther, N. W. Sochacka, and N. N. Kellam, "Quality in Interpretive Engineering Education Research: Reflections on an Example Study," *J. Eng. Educ.*, vol. 102, no. 4, pp. 626–659, Oct. 2013, DOI: 10.1002/jee.20029.
[41] S. Secules, A. Gupta, A. Elby, and C. Turpen, "Zooming Out from the Struggling Individual Student: An Account of the Cultural Construction of Engineering Ability in an Undergraduate Programming Class.," *J. Eng. Educ.*, vol. 107, no. 1, pp. 56–86, Jan. 2018, DOI: 10.1002/jee.20191.
[42] E. Soloway and J. C. Spohrer, Eds., *Studying the Novice Programmer*, 0 ed. Psychology Press, 2013. DOI: 10.4324/9781315808321.
[43] E. Soloway, "Learning to program = learning to construct mechanisms and explanations," *Commun. ACM*, vol. 29, no. 9, pp. 850–858, Sep. 1986, DOI: 10.1145/6592.6594.
[44] M. van den Haak, M. De Jong, and P. Jan Schellens, "Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue," *Behav. Inf. Technol.*, vol. 22, no. 5, pp. 339–351, Sep. 2003, DOI: 10.1080/0044929031000.
[45] *iMotions*. [Online]. Available: https://imotions.com/platform/
[46] "Shimmer Wearable Sensor Technology | Wireless IMU | ECG | EMG | GSR," *Shimmer Wearable Sensor Technology*. https://shimmersensing.com/ (accessed May 15, 2022).
[47] "Aurora," *Smart Eye*. https://smarteye.se/research-instruments/aurora/ (accessed May 15, 2022).
[48] E. Wert, J. Grifski, S. Luo, and Z. Atiq, "A Multi-Modal Investigation of Self-Regulation Strategies Adopted by First-Year Engineering Students During Programming Tasks," in *Proceedings of the 17th ACM Conference on International Computing Education Research*, Virtual Event USA, Aug. 2021, pp. 446–447. DOI: 10.1145/3446871.3469795.