A Fine-grained Assessment on Novice Programmers' Gaze Patterns on Pseudocode Problems

Unaizah Obaidellah
University of Malaya
Kuala Lumpur, Malaysia
unaizah@um.edu.my

Tanja Blascheck
University of Stuttgart
Stuttgart, Germany
Tanja.Blascheck@vis.uni-stuttgart.de

Drew T. Guarnera
Kent State University
Kent, USA
dguarner@kent.edu

Jonathan I. Maletic
Kent State University
Kent, USA
jmaletic@kent.edu

ABSTRACT

To better understand code comprehension and problem solving strategies, we conducted an eye tracking study that includes 51 undergraduate computer science students solving six pseudocode program comprehension tasks. Each task required students to order a sequence of pseudocode statements necessary to correctly solve a programming problem. We compare the viewing patterns of computer science students to evaluate changes in behavior while participants solve problems of varying difficulty. The intent is to find out if gaze patterns are similar prior to solving the task and if this pattern changes as the problems get more difficult. The findings show that as the difficulty increases regressions between areas of interest also tend to increase. Furthermore, an analysis of clusters of participants' common viewing patterns was performed to identify groups of participants' sharing similar gaze patterns prior to selecting their first choice of answer. Future work suggests an investigation on the relationship of these patterns with other background information (such as gender, age, English language proficiency, course completion) as well as performance (score, duration of task completion, competency level).

CCS CONCEPTS

• **Human-centered computing** → **Visualization**; • **Theory of computation** → **Program schemes**; • **Applied computing** → **Education**.

KEYWORDS

program comprehension, eye tracking, human-computer interface, teaching/learning strategies, improving classroom teaching, programming education

1 INTRODUCTION

The ability to solve problems effectively is one of the key determinants for programmers to write good computer programs. At present, there is still lack of work in identifying a programmers' state of comprehension as an indicator of their underlying mental model especially with the use of data mining and machine learning techniques. It is not trivial for programmers to externalize their mental model while they are doing a task. Tracking developers' eyes while they are doing a task helps to determine what they were looking at while solving the task. Such insights help in constructing a mental model of programmers without explicitly asking them what they looked at. In the area of programming education, a deeper understanding about students' cognitive processes and cognitive framework is crucial to help educators in designing improved teaching and learning strategies and instructional design models.

In recent years eye tracking has been utilized by researchers to study the cognitive process of students during programming activities [Lai et al. 2013; Obaidellah et al. 2018; Sharafi et al. 2015]. Besides mapping the use of eye trackers in evaluating students' performance, behavior and characteristics of novice and experts in recent research, Obaidellah et al. [2018] also identified the lack of studies on some common programming education tools such as pseudocodes. In an eye tracking study, Andrzejewska et al. [2016] compared students' attention and approaches in pseudocode and flowcharts, and reported a relationship between high performers and their close attention on pseudocode presentations, whereas low performing students preferred and attended more on flowchart presentation.

To investigate how undergraduate computer science students read and understand a code problem presented as pseudocode we designed and conducted an eye tracking study using six pseudocode problems of varying difficulty (easy, medium, difficult) to evaluate how similar 51 students' gaze patterns are when solving these problems to find common comprehension strategies. Our goal is to
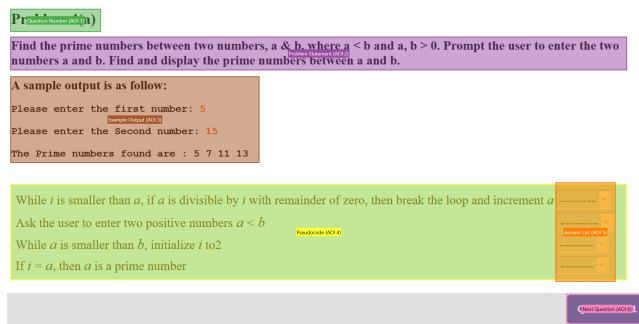
Figure 1: Example of a medium difficult stimulus with AOIs shown as colored rectangles. The general structure of the stimulus is: heading (green, AOI 1), problem statement (purple, AOI 2), followed by an output example (brown, AOI 3), pseudocode statements (yellow, AOI 4), an answer selection list (orange, AOI 5) and the next button (pink, AOI 6).

inspect the gaze patterns of students when solving programming problems presented as pseudocode. We are especially interested in finding patterns using the apriori algorithms before students input their first answer (i.e., their first mouse click) and patterns that occur before moving on to the next problem. We also use hierarchical clustering to find groups of participants based on these patterns. The main contribution of this work are the exploration of methods used to analyze eye movement data to 1) identify participants' common gaze patterns (i.e., AOI sequences) for a portion of a problem solving task before they input their first as well as final answer, and 2) find groups of gaze patterns that share similarity among undergraduate computer science students.

2 STUDY DESIGN

We conducted an eye tracking study to investigate the following research questions:

- RQ 1: Are students' gaze patterns prior to entering the first answer and before entering the final answer similar?
- RQ 2: Do students' gaze patterns to comprehend pseudocode problems change and become more diverse as problems become more difficult?

We are motivated to better understand if there exist, among the participants, a common problem solving pattern in terms of the process of validating (if any) their comprehension (first answer) and answers (if any) before an action (mouse click submission) is made and whether this is affected by the question difficulty. This would inform about the degree of confidence exhibited by these programmers during a problem solving task. If an identical pattern is found across all participants, it is likely that the students regardless of experience and expertise may potentially adopt similar validation strategy and confidence in confirming their understanding and answer choices. Furthermore, it is estimated that students' gaze patterns would change as the problems become difficult.

Table 1: The programming problems presented to the participants with difficulty level and lines of code (LOC).

Level	Problem	LOC
Easy	Random number	4
	Print reverse sentence	4
Medium	Recursive function call	5
	Prime numbers	4
Hard	Count characters from file	6
	Binary search	9

2.1 Materials

Four course instructors selected, reviewed and ranked a set of six programming problems as easy, medium, and hard in terms of their difficulty level. Figure 1 shows an example of a medium difficult stimulus. The steps of the solution in the form of pseudocode statements are given in random order and participants saw the same set of stimuli in the same order. In total, participants have to complete six problems detailed in Table 1, which are presented in a web browser integrated into the eye tracking device. There is no time restriction to complete each problem.

2.2 Participants

Fifty-six first year CS-major undergraduate participants aged between between 19 and 23 years old (M_{age} = 20.7 years, SD_{age} = 1.1) took part in the study via on-campus mailing advertisements. Participants had varying experience with programming and pseudocode but all were familiar with the types and format of the problems given their similarity with those taught to them. They completed a post-study survey at the end of the tasks. Data from five students from the pool had to be discarded due to data collection issues.

2.3 Procedure

The Tobii T120 eye tracker was used to record the eye movements. The study was conducted in a dedicated office space at the faculty's building during a two-week period. Participants were individually allocated a specific time slot of one hour for the study. After calibration, participants were left alone to go through all sets of problems individually with no interaction or interference from the experimenter. The given task was to rearrange the random pseudocode statements in a correct order for each problem. All students were compensated a food voucher at the end of the study. We exported all data including fixations, areas of interest (AOIs), and mouse click events for further analysis.

2.4 Areas of Interest and Mouse Clicks

For each stimulus we defined 6 AOIs, which are shown in Figure 1. The general structure for AOIs on each stimulus is: heading (AOI 1), problem statement (AOI 2), followed by an output example (AOI 3), pseudocode statements (AOI 4) an answer selection list (AOI 5) and the next button (AOI 6). The fixation data of participants were assigned to one of these 6 AOIs. Note that AOI 4 and AOI 5 overlap. Therefore, it is possible for fixations to be mapped to multiple AOIs. Along with AOI a participant's mouse clicks are captured while solving each problem. We use mouse click events to group AOIs

into sets, e.g., AOI sequences that occur before participants interact with a stimulus for the first time or before proceeding to the next problem.

3 STUDY RESULTS

We analyze the eye movement data collected during the study using AOI sequences to find common patterns. First, we analyze the AOI sequences that occurred before the first mouse click as well as the AOI sequence before participants move on to the next problem. In addition, we cluster these AOI sequences to find if similar gaze-patterns appear among the participants.

3.1 Patterns of AOI Sequences

To identify potential patterns in the AOI sequences, the apriori algorithm provided by the arules library in R is used. We limit the patterns to the first set of AOI sequences before the first click and the last set of AOI sequences before participants click on the button to submit their answers. Because patterns in between the first and last AOI sequences can vary due to the nonlinear nature of ordering the pseudocode steps, we assume the first and last sequences to contain landmark points in the task in which students may share similar fixation behaviors. To refine the granularity of each pattern, subsets are generated containing the last three, four, and five elements in each AOI sequence.

Running the apriori analysis reveals some consistent patterns and rules within AOI sequences before the first click event of a task. Interestingly, the AOI sequence before the last click event provides no discernible rules or patterns at all. This indicates that some consistent fixation activities occur for initial task comprehension, but after that point, the fixation behavior of the students tends to diverge to a strong degree. From here we narrow the Apriori analysis to just the first set of AOI sequences for each problem and their associated subset sizes. Initial rules produced indicate that fixations on the pseudocode section (AOI 4) and a combined fixation between the pseudocode section and the answer input section (labeled AOI 4+AOI 5) are expected to appear in nearly every pattern with high support and confidence. This finding is intuitive and occurs across all length AOI subsets (three, four, and five) as these sections required to successfully complete a task and are expected to be viewed most frequently. However, the lift value of approximately 1 for the rule indicates that AOI 4 and AOI 5 are present in most rules due in part to their prolific occurrence. There is a slightly stronger pattern between the occurrence of AOI 4 with a combined fixation on AOI 4+AOI 5 and vice versa with a lift of 1.1 indicating a tendency for participants to examine the answer section (AOI 5) when referring to the pseudocode steps (AOI 4).

Itemset mining analysis is also applied to the AOI sequences after converting them to a scan path format. This requires the removal of any consecutive duplicate values from each AOI set and provides a simplified representation of the participants viewing patterns. Once again, sequences are divided into subsets representing the last three, four, and then five AOIs present before the click event. While AOI 4 and the combined AOI 4+AOI 5 fixations still dominate the rules with negligible lift (approximately 1), a few more subtle patterns of interest emerge. Examination of the scan path subset of size three on problem P 1, P 3, and P 5 show that there is a tendency

for participants who viewed the output section of the task (AOI 3) to not also view the pseudocode section (AOI 4) indicated by lift values between .93 and .99. These rules, however, are still supported with around 50% of the data and a high confidence around 90%. Increasing the subset length of the scan paths to include the last four elements, the previous rule continues to hold with similar support, confidence, and lift for problem P 1 and P 5 and emerges as a new rule for P 4. Extending the sequence once again to five elements shows that the rule still appears but only for problem P 1 and P 4. This rule is interesting in that it indicates some participants seem to focus more on the details of the problem independent of the pseudocode statements before their first click event.

One other pattern only found in scan path subsets of size four or larger is the presence of rules involving the problem statement (AOI 2). In these sequences we find rules indicating between 40% and 60% of participants that view the problem statement (AOI 2) tend to be less inclined to have fixations near the answer section of the task (AOI 4+AOI 5 and AOI 5) with lift values between .90 and .99. It is possible that this rule indicates a top down review of the problem details and output section before attempting to select answers to the problem. With many item set rules having either low support, confidence, or subtly significant lift values, a broad analysis of fixation regressions was examined to identify viewing patterns.

3.2 Regression Patterns

Recall that all fixations are collected before the first click event are recorded. Fixation regressions are totaled based on the number of nonconsecutive recurrent views for each fixation in the sequence. Examining the total number of regressions for each of the six problems shows a general increase in the number of fixation regressions as the difficulty of the problem increases. The only problem in which this pattern does not hold is problem P 2, the second easiest problem in the set, which has the lowest number of regressions (540) among all problems. Because problem P 2 focuses on the manipulation of a short string, it is possible that participants are more familiar with this type of problem resulting in less fixation transitions from the pseudocode instructions (AOI 4) to the problem statement (AOI 2) or expected output sections (AOI 3).

3.3 Hierarchical Cluster Analysis

We continued our assessment using hierarchical cluster analysis (HCA) to identify the number of clusters indicating participants' tendency of sharing similar gaze patterns prior to selecting their first choice of answer. The HCA is a form of a classification technique that produces a set of clusters containing participants' gaze patterns which are similar to each other while preserving differences between the clusters. A pre-specification of the number of clusters to be generated is not required for this analysis. Adopting the agglomerative (AGNES) type (i.e.,bottom-up approach that begins with each case in a separate cluster and increasingly combines clusters until only one is left) and squared Euclidean distance in determining the distance (or dissimilarity) of two clusters, produced clusters of size 10, 10, and 6 respectively for the easy, medium and hard problems. The dendrogram for the hard problem is shown in Figure 2. The optimal number of cluster is calculated using the
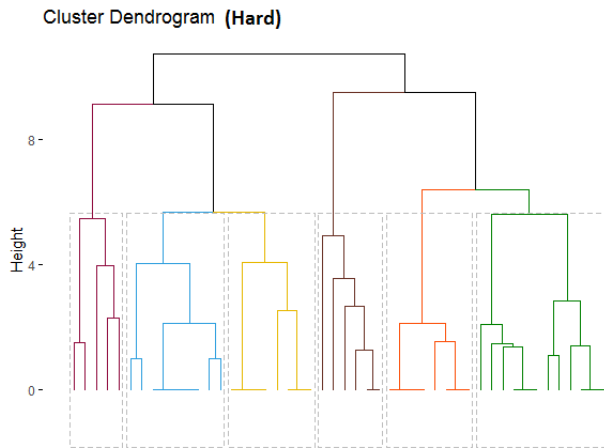
Figure 2: Dendrogram for hard problems with number of cluster, k = 6. Each unique color and bounding box denote a single cluster.

NbClust library in R based on majority rule principle by comparing across varying combinations of number of clusters, distance measures and clustering methods. This result can be interpreted as a large variety of gaze patterns which occur for each type of problem, especially for the easy and medium problems. Furthermore, it is worth noting that when evaluating the clustering tendency, the Hopkins statistic, H = 0.33 for easy, H = 0.30 for medium, and H = 0.38 for hard problems suggest that the dataset is less likely clusterable or not uniformly distributed. Taken together, this gives an indication that no gaze pattern (single cluster) appears stronger than the others with regard to the problem difficulty. Again, these results support the previous findings.

4 DISCUSSION

RQ1: Are students' gaze patterns prior to entering the first answer and final answer similar? As we will see, we found that the gaze patterns the students exhibited for the first and final answers are dissimilar. Reflecting our hypothesis, these students were probably using different validation strategies with varied confidence levels. Our findings from the apriori algorithm analysis (to generate frequent itemset and to find association rules) indicated that gaze on the pseudocode section and answer lists were found to emerge across all problems for almost all participants' only moments before a first click event occurs. These expected patterns happen most frequently given that participants' needed to read and identify the pseudocode statements to match the ordering with the correct answers from the drop-down list option. This finding supports those reported by Obaidellah and Haek [2018].

In terms of identifying which AOIs participants generally focused on to understand the problems, the scanpath analysis of the last three, four and five AOIs before the first click event for a majority of the problems reports a higher tendency of participants' visual attention focusing on the output example (AOI 3) compared to the pseudocode statement (AOI 4) or the problem statement (AOI 2). This means, the output example has a more critical role

in shaping the students' understanding compared to the textual problem statement. This could be influenced by the more simplified form of the problem represented as an expected solution shown in the output example than those represented in textual form. It is also likely that the participants are more comfortable to derive the problem requirements from an example solution. Indirectly, this result gives a recommendation that providing a sample output for a programming task is beneficial to assist students in program comprehension.

Once again, referring to the apriori and scanpath analyses, a similar pattern is not evidently representative prior to the participants' click on the answer submission button to proceed with the following question. A potential reason for the existence of diverse patterns before entering the final answer (i.e., click on the submit button) is likely due to different validation strategies (if any) participants adopted.

Given that the tested problems varied in terms of difficulty—easy, medium and hard—we also assume that the gaze patterns of the problem solvers would change accordingly with an easier question showing a more consistent pattern, and increasingly less consistent for more difficult problems. However, our observations show otherwise. The easy and medium problems have more diverse patterns than the difficult problems as the cluster analysis revealed. This could mean that the participants' solution and comprehension strategies diverge across the problems, within and between difficulty levels. This could remain true for the easy and medium problems. However, as the problems become more difficult, the gaze patterns were found to be more consistent for all students. Thus, maintaining a smaller degree of difference between the problems (difficult) in terms of the gaze patterns. Thereby, producing a smaller number of clusters.

Another potential explanation for this result is that it is probable that the participants' viewed the difficulty of the problems differently than the experimenters intended. This possibility holds a good chance given that the participants who are novice programmers (undergraduate computer science students) are equipped with a wide degree of experience (knowledge) and skills. Hence, it is unsurprising that the students' process of problem-solving strategy and level of comprehension varies. Consequently, poorer performing students of a task who demonstrated higher cognitive load (high amount of mental effort or working memory resource) that requires deeper mental processing in comprehending and organising solution strategies could have adopted certain unique strategy that singles them out from forming clusters of gaze patterns similar with other higher-performing students who may potentially possess a more organized mental schema and a well-defined solution strategy [Bergersen and Gustafsson 2011; Paas and Van Merriënboer 1994]. These claims require further validation. Therefore, it is important to replicate this study with other types of stimuli of different difficulty to confirm the current findings.

RQ2: Do students' gaze patterns to comprehend pseudocode problems change and become more diverse as problems become more difficult? In terms of the sequence of AOIs visited, the gaze patterns for the problems remain reasonably similar, even as the difficulty levels increase. This implies that the problems posed represented the expected level of complexity. For example, there was a larger

number of lines of pseudocode for the harder problems. Given the limited capacity of the working memory [Cowan 2010] that applies to typical adults at a particular duration, it is reasonable to acknowledge that regular regressions would take place in assisting comprehension, solution strategies and validation of answers (if any). Therefore, consistent with our hypothesis, it is further expected that our findings showed that regression counts increase with the question difficulty (except for P 2).

5 CONCLUSIONS AND FUTURE WORK

The analysis reported in this paper evaluates the last three AOIs students view prior to inserting their first answer. We adopted data mining (the apriori algorithm) and machine learning (HCA) techniques in analysing our data. It is noted that although the findings are less conclusive, the proposed method has a potential for further exploration. We propose that a closer inspection on HCA considering its relationship with participants demographics (gender, year of study and English proficiency) and performance data (accuracy, task completion duration and competency level).

ACKNOWLEDGMENTS

REFERENCES

Magdalena Andrzejewska, Anna Stolińska, Władysław Błasiak, Paweł Pęczkowski, Roman Rosiek, Bożena Rożek, Mirosława Sajka, and Dariusz Wcisło. 2016. Eye-tracking verification of the strategy used to analyse algorithms expressed in a flowchart and pseudocode. *Interactive Learning Environments* 24, 8 (2016), 1981–1995.

Gunnar Rye Bergersen and Jan-Eric Gustafsson. 2011. Programming skill, knowledge, and working memory among professional software developers from an investment theory perspective. *Journal of individual Differences* (2011).

Nelson Cowan. 2010. The magical mystery four: How is working memory capacity limited, and why? *Current directions in psychological science* 19, 1 (2010), 51–57.

Meng-Lung Lai, Meng-Jung Tsai, Fang-Ying Yang, Chung-Yuan Hsu, Tzu-Chien Liu, Silvia Wen-Yu Lee, Min-Hsien Lee, Guo-Li Chiou, Jyh-Chong Liang, and Chin-Chung Tsai. 2013. A review of using eye-tracking technology in exploring learning from 2000 to 2012. *Educational research review* 10 (2013), 90–115.

Unaizah Obaidellah, Mohammed Al Haek, and Peter C-H Cheng. 2018. A survey on the usage of eye-tracking in computer programming. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 5.

Unaizah Obaidellah and Mohammed Al Haek. 2018. Evaluating gender difference on algorithmic problems using eye-tracker. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, ETRA 2018, Warsaw, Poland, June 14-17, 2018.* 2:1–2:8.

Fred GWC Paas and Jeroen JG Van Merriënboer. 1994. Instructional control of cognitive load in the training of complex cognitive tasks. *Educational psychology review* 6, 4 (1994), 351–371.

Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. 2015. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology* 67 (2015), 79–107.