Classification of Strategies for Solving Programming Problems using AoI Sequence Analysis

Unaizah Obaidellah, Michael Raschke, Tanja Blascheck

Classification of Strategies for Solving Programming Problems using AoI Sequence Analysis

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

Michael Raschke
Blickshift Gmbh
Stuttgart, Germany
michael.raschke@blickshift.de

Tanja Blascheck
Inria
Saclay, France
tanja.blascheck@inria.fr

ABSTRACT

This eye tracking study examines participants' visual attention when solving algorithmic problems in the form of programming problems. The stimuli consisted of a problem statement, example output, and a set of multiple-choice questions regarding variables, data types, and operations needed to solve the programming problems. We recorded eye movements of students and performed an Area of Interest (AoI) sequence analysis to identify reading strategies in terms of participants' performance and visual effort. Using classical eye tracking metrics and a visual AoI sequence analysis we identified two main groups of participants—effective and ineffective problem solvers. This indicates that diversity of participants' mental schemas leads to a difference in their performance. Therefore, identifying how participants' reading behavior varies at a finer level of granularity warrants further investigation.

CCS CONCEPTS

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

KEYWORDS

Eye tracking, visual analytics, reading strategies, programming problems

ACM Reference Format:
Unaizah Obaidellah, Michael Raschke, and Tanja Blascheck. 2019. Classification of Strategies for Solving Programming Problems using AoI Sequence Analysis. In *2019 Symposium on Eye Tracking Research and Applications (ETRA '19), June 25–28, 2019, Denver , CO, USA.* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3314111.3319825

1 INTRODUCTION

One of the key factors for the success of writing computer programs is the ability to solve problems [Détienne and Soloway 1990]. At least among novice computer science students, effective strategies that potentially guide students towards reaching efficient solutions is important to maintain their motivation to learn and increase their computer programming skills. Often, in a classroom setting, students vary in terms of expertise and individual factors such as personal traits, experience, and cognitive ability, which leads to different problem solving strategies [Sharma et al. 2018].

One of the abilities required for problem solving is the use of previous knowledge. In *Schema Theory* [Détienne and Soloway 1990] previous knowledge is defined as the past experience acquired when solving similar tasks. However, as Gomes and Mendes [2007] indicated, poor problem solving abilities due to the inability to establish correct analogies with past problems and poor skills at transferring previous knowledge to new problems can cause difficulties to solve programming problems. The ability of computer science students to bring past knowledge into the working memory to formulate a solution to a new programming problem determines their level of expertise. Therefore, students with greater expertise have a higher chance of considering multiple aspects of a programming problem to arrive at a reasoned response than novices.

Hence, students achieve different levels of success in writing effective computer programs. These different learning strategies introduce the challenge that educators need to identify those students needing help early and quickly. The outcome of this early detection should support and improve teaching interventions because students can then be given the appropriate attention depending on their level of performance. Therefore, gaining a comprehensive understanding in examining how students use their mental schema in solving fundamental computer programming exercises is critical to propose appropriate guidance.

To study and detect different levels of performance, eye tracking is a suitable technology. Eye tracking allows to assess peoples' reading behavior and objectively indicates their underlying cognitive processes [Meng Lung et al. 2013; Stolińska and Andrzejewska 2017]. Therefore, this paper contributes an eye tracking study that examines student's visual attention and mental schema when solving programming problems. The outcome of this work gives a first hint on problem solving strategies of novice programmers based on the authors' personal interpretation of the data using visual analysis methods that raises further important research questions for follow up research.

2 RELATED WORK

Past research on schema theory in programming [Duran et al. 2018; Schulte et al. 2010; Soloway 1986] has investigated program construction and comprehension involved while participants solve programming problems. Détienne and Soloway [1990] proposed a model of mental schema to identify expert's comprehension strategies of common and uncommon (plan-like versus unplan-like) computer programs of a fill-in-the-blanks task. The authors analyzed

ETRA '19, June 25–28, 2019, Denver , CO, USA

ACM ISBN 978-1-4503-6709-7/19/06…$15.00
https://doi.org/10.1145/3314111.3319825

this using a verbal protocol approach. Four comprehension strategies were derived following the level of expectation of the programmer. Rist [1989] extended Soloway's exhaustive work [Détienne and Soloway 1990; Soloway 1986; Soloway et al. 1983; Soloway and Ehrlich 1984; Spohrer et al. 1985] to show the effect of schemas on programming strategies. It was found that novices and experts program in a top-down manner if the problem is consistent with their mental schema, but revert to constructing solutions in a bottom-up approach when their mental schemas are incomprehensive for the problem considered. These findings have primarily focused on assessment of source code deduced from non-eye-tracking methods. To our knowledge, there exist no well-established studies and methods that have investigated problem solving for programming to identify schemas based on eye tracking. The earliest paper using eye tracking to study programming comprehension was reported by Crosby and Stelovsky [1990]. A recent comprehensive survey conducted by Obaidellah et al. [2018] reported that this popular research topic records the highest number of eye tracking research papers published up to date. Evaluating how people understand programming problems and identifying solutions well before writing code is deemed important because this sets the initial phase for writing a successful program.

To analyze the collected eye movement data, we use a visual analysis method. Different visualization techniques for eye movement data exist [Blascheck et al. 2017], which are often superior to the classical attention maps [Holmqvist et al. 2003, Chapter 7] and gaze plot [Holmqvist et al. 2003, Chapter 8]. On one hand, standard attention maps illustrate a general distribution of fixation after the completion of a task giving an overview, however, analyzing the actual sequence of fixations is not possible. On the other hand, gaze plots showing the sequence of fixations become overcrowded for long tasks and many participants. Therefore, we use scarf plots [Richardson and Dale 2005], a method based on areas of interest (AoIs). Scarf plots aggregate the fixations based on AoIs and shows the duration of each AoI visit in a unique color for each AoI. This enables us to closely inspect the order of AoIs and how long they have focused on each AoI.

3 STUDY

Mental schema can be described in two aspects, as knowledge to assist program understanding (declarative) and as cognitive mechanism to use the knowledge (procedural). In this work, the knowledge aspect assumes students to have a 'solution plan'—fragments that represent typical action sequences in solving programming problems. Hence, characteristics such as the number of variables, data types, and operations are considered common components that form the solution plan of a problem. During problem solving, information extracted from the problem statement posed, enables the activation of mental schema. The evoked knowledge creates expectation on what information should be available in the solution. Consistent with the strategies proposed by Détienne and Soloway [1990], solving a programming problem requires constructing a representation of the problem that consists of plans and goals. This structure or mental schema used to solve problems represents an explanation of what the solution does and how it is achieved. The mental schema enables the problem solver to predict which type of suitable mechanism should be used in solving different problems. To understand computer science students' underlying mental schema when solving programming problems in the form of problem statements we conducted an eye tracking study. In the following sections we introduce the research questions, the study design and tasks, the participants, the technical setup, and the procedure we followed.

3.1 Research Questions

We designed the study to answer the following research questions:
Research Question 1. How do students comprehend and solve programming problems?
Research Question 2. Which classification of reading strategies for programming problems can we derive?

3.2 Study Design and Tasks

Each participant attempted to solve a set of nine programming problems, ranging from easy, medium, and hard in terms of their difficulty level with three problems for each difficulty level. The set of questions according to their difficulty level were adopted from a textbook used for a *Fundamental of Computer Programming* course and three computer science academics who have experience teaching computer programming at the undergraduate level further validated the difficulty level.

The programming problems were designed for participants to read a problem statement, examine an example output provided, and answer four questions relating to the problem (cf. Figure 1). The first three questions, respectively, required the participant to identify 1) the number of variables that should be defined for the problem in consideration, 2) the data types that best suit the chosen variables, and 3) the operations best used to solve the problem. The fourth question asked participants to rate the level of difficulty of the programming problem. The sequence of these questions remained the same for all nine stimuli. The students were familiar with the types of problems presented as the requirements, nature, and format of the questions were similar to those taught and presented to them in their classroom exercises.

We divided each stimulus into six main AoIs (cf. Figure 1): problem statement (PS), example output (OP), variable(s) (Var), data type(s) (DT), operation(s) (Oper), and difficulty level (Diff). Participants had to answer the multiple-choice questions by selecting one or multiple answers. The levels of difficulty of the programming problems were easy (revenue calculation; leap year; random number multiplication), medium (Body Mass Index (BMI) calculation; palindrome; scissors, rock, paper game), and difficult (occurrence of the largest number; reverse an integer; geometry: point in a circle).

3.3 Participants

Thirty six undergraduate students of the Computer Science major from a large public university in Southeast Asia responded to a call for participation posted through a campus mailing list. We specifically asked for participants who had completed their final exam of the *Fundamental of Computer Programming* course at the end of semester one of their academic year. Therefore, the study took place at the beginning of semester two. At the time of the study, nine of the participants were retaking the course as they received scores below standard course passing rate (<50%) in their

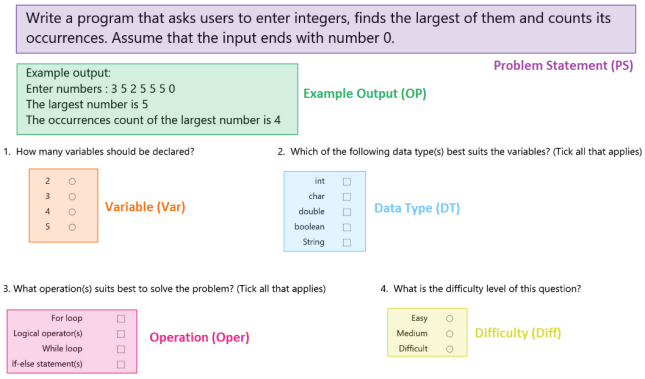

Figure 1: Example of a difficult stimulus with AoIs shown as colored rectangles. The general structure of the stimulus is: problem statement, followed by an example output, then four multiple-choice questions regarding 1) the variables that have to be declared, 2) the data types, 3) the operations, and 4) the difficulty of the question.

first attempt. The participants were between 19 and 24 years old (M_{age} = 21, SD_{age} = 1.2), with 22 female and 14 male. The participants identified themselves as Asians with good level of English proficiency. None of the participants' data was discarded.

3.4 Technical Setup

In this study, the GP3 Gazepoint eye tracker with a sampling rate of 60 Hz and 0.5 degree of accuracy was used with a 23" monitor and full HD resolution of 1920 × 1080 pixels on a desktop computer. All eye movement data was recorded using the iMotions software. The stimuli were prepared using the Survey Slide module of iMotions with a resolution of 1920 × 1114 pixels. The stimuli size was automatically down-scaled to the monitor's resolution during the recording. Prior to further analysis, the recorded eye movement data was transformed to fixation data using the dispersion algorithm [Salvucci and Goldberg 2000] with the following parameter values: maximum dispersion = 18, minimum rows = 5.

We conducted a 9-point calibration for each participant before they began their task. We exported all the raw data from the iMotions software as a text file and used Blickshift Analytics to map fixations to AoIs. The stimuli, AoIs, raw fixation data, and questionnaires of the study are added as supplemental material.

3.5 Procedure

Participants arrived individually to a dedicated office space in the faculty's building at a scheduled time. Upon arrival, each participant was seated at about 60 cm distance in front of the eye tracker placed below the computer's monitor. The participants were reminded to sit comfortably. Once participants indicated their readiness to begin the task, their eyes were calibrated. Then, participants were left alone to read the instructions of the task prior to watching a short video showing an example of the programming problems posed and the way to navigate through the programming problems. Prior to the actual test, participants completed an example programming problem to familiarize themselves with the programming problems presented. The order of the nine programming problems were randomized for every participant to avoid order effects. The tasks occurred without any interference or interaction with the experimenter. There was no time limit to each session. However, participants were reminded to perform the task to their best ability in the shortest time possible. Upon completion of the tasks, participants completed a post study survey, in which they indicated their programming experience and preferences. In total, each session lasted an average of 40 minutes, starting from arrival time to post study survey. Each student received a food voucher worth a single meal as a form of compensation at the end of their session.

4 RESULTS AND ANALYSIS

We analyze our eye movement data using quantitative and qualitative methods. First, we discuss an initial grouping of participants based on their performance in an exam, then we re-group participants based on their performance in the actual study. We use scarf plots [Richardson and Dale 2005], an AoI sequence analysis, to identify participants' reading patterns according to their performance in solving algorithmic questions.

4.1 Participant Groups

We initially analyzed the data by separating the participants into high and low performing students. The high performing group consisted of participants who scored grade A (≥ 80%) in the *Fundamental of Computer Programming* course, while the low performing group consisted of students who scored grade C (≤ 50%). However, analysis of each group based on their completion time and accuracy yielded inconsistent results in terms of participants' reading strategies. The mean accuracy calculated across all questions showed that all participants scored above 50%, limiting the characteristics to group the low and high performing students based on their grades.

The second approach was to group participants according to their average completion time across all nine stimuli in relation to their accuracy. We classified participants based on two time categories (short completion time and long completion time). As a threshold we used the average completion time for all participants and all stimuli ($Avg.$ = 204.5s). We also calculated the accuracy for each participant, which is the percentage of correctly answered stimuli. Because each stimulus consisted of three multiple-choice questions, participants had to have 50% (2 out of 3) of them correct for the stimulus to count as correctly answered. Although multiple answers in each multiple-choice question could be counted as correct (i.e., choosing a while-loop instead of a for-loop), we only counted an answer as correct if a participant chose a correct answer for the problem at hand. Because the participants in the study were students from a beginners programming course, it was an objective of the course to teach students to choose the appropriate variables and constructs (i.e., using a double only if this is required). This gives us four accuracy groups: an accuracy score of 56% (5 out of 9 stimuli correct), 67% (6 out of 9), 78% (7 out of 9), and 89% (8 out of 9). The initial two accuracy scores are categorized as low scores, while the latter two as high scores. The combination of these two metrics, results in a 2 × 2 matrix (i.e., time × score). This classification returns four participant groups (short&low (SL); short&high (SH); long&low (LL); long&high (LH), cf. Table 1). Subsequently,

Table 1: Name and number of participants (in parentheses) in each of the four groups assigned based on their completion time and accuracy score.
Legend: short&low (SL); short&high (SH); long&low (LL); long&high (LH).

	Low Accuracy Score	High Accuracy Score
Short Completion Time	SL (11)	SH (8)
Long Completion Time	LL (8)	LH (9)

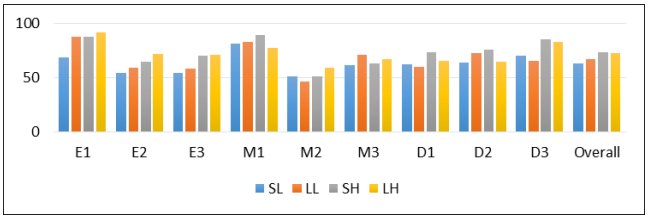

Figure 2: Percentage of accuracy score across participant groups for each stimulus. SH and LH scores are higher than the scores in the low groups in E2, E3, M1, D1 and D3.
Legend: easy (E); medium (M); difficult (D); short&low (SL); short&high (SH); long&low (LL); long&high (LH).

the reading behavior of these four participant groups was analyzed quantitatively and qualitatively across all stimuli.

4.2 Descriptive Measures

At an aggregated level, participants in the short (SL, SH) groups ($M = 150, SD = 38$) spent less time completing the tasks than those in the long (LL, LH) groups ($M = 265, SD = 57$). In terms of accuracy, the low (SL, LL) groups ($M = 65, SD = 5$) recorded lower scores than the high (SH, LH) groups ($M = 73, SD = 6$). To evaluate how much participants' ratings were consistent with the level of difficulty of the programming problems we analyze the answers. The analysis shows that most stimuli were rated easy, receiving highest rating in decreasing order of LH ($M = 3.00, SD = 1.22$), SH ($M = 2.25, SD = 0.71$), SL ($M = 1.82, SD = 0.98$), and LL ($M = 1.38, SD = 0.52$). Rating as medium difficult for the questions were lower than those of easy with mean values recorded almost equal for all groups, LH ($M = 0.89, SD = 1.05$), SH ($M = 0.88, SD = 0.83$), SL ($M = 1.00, SD = 0.63$), LL ($M = 1.13, SD = 0.99$). Only the low scoring groups (SL, LL) indicated some of the stimuli as difficult. The SL group ($M = 0.45, SD = 0.69$) rated the difficulty slightly higher than the LL group ($M = 0.13, SD = 0.35$). These ratings suggest that the high performing groups (SH, LH) were more confident with their performance compared to the low performing groups (SL, LL). Additionally, as shown in Figure 2, the high performing groups (SH, LH) achieved better scores than the low performing groups (SL, LL) in five out of nine stimuli.

4.3 Fixation Metrics

In addition to the task completion time and accuracy scores, we analyze standard eye tracking metrics (fixation duration, fixation count) to highlight the main differences in the way participants from the different groups read, comprehended, and solved the programming

Table 2: Average fixation duration (sec.) for all stimuli for all participant groups. The bold values indicate longer fixation duration spent on the respective AOIs when comparing time for equally performing groups (SH vs LH, SL vs LL).
Legend: short&low (SL); short&high (SH); long&low (LL); long&high (LH); easy (E); medium (M); difficult (D); problem statement (PS); example output (OP); variable(s) (Var); data type(s) (DT); operation(s) (Oper).

SL	E	M	D	Mean
PS	126	123	94	**115**
OP	165	135	143	148
Var	83	106	92	94
DT	126	127	104	**119**
Oper	148	144	123	138

SH	E	M	D	Mean
PS	96	101	73	**90**
OP	171	133	177	**160**
Var	67	71	74	71
DT	105	104	83	97
Oper	126	122	125	124

LL	E	M	D	Mean
PS	114	118	93	108
OP	205	161	187	185
Var	87	93	90	90
DT	120	118	97	112
Oper	146	147	125	139

LH	E	M	D	Mean
PS	88	84	65	79
OP	124	130	131	128
Var	85	97	84	88
DT	132	112	122	122
Oper	138	130	124	131

problems. Due to limited space for reporting and complexity of analysis across stimulus difficulty level (easy (E), medium (M), and difficult (D)), we only report the group effects (SL, SH, LL, LH).

4.3.1 Fixation duration. The average fixation duration for all AoIs across stimuli were calculated to find if the participant groups spent a different amount of time on different parts of the programming problem. Table 2 reports the mean fixation duration for each participant group. Comparing the time groups (short versus long) for the equally performing groups (SH versus LH, SL versus LL), we observe that the SH group spent a longer time on AoIs *PS* and *OP*, while the opposite is true for the LH group. The SL group spent more time on AoIs *PS* and *DT* but less time on *OP* compared to the LL group. However, the SL and LL groups spent almost the same amount of time on *Var* and *Oper* for the programming problems examined. Between the score groups of similar duration (SL versus SH, LL versus LH), the SL group spent more time than the SH group across all AoIs (except for *OP*). The mean data also shows that the SL group has a larger time range than the SH group. Participants in the LL group spent more time on *PS*, *OP*, and *Oper*, while the LH group generally spent slightly more time on *DT*. The long groups showed almost equal amounts of time on the *Var*.

A closer inspection of the fixation duration on each AoI shows that all participant groups showed similar patterns of decreasing time in the order of *OP*, *PS*, *Oper*, *DT*, and *Var*. This is confirmed by the non-parametric test (Friedman's ANOVA) analyzed for each participant group. The fixation duration significantly changes across all AoIs: SL [$\chi^2(4) = 127.9, p < .001$]; SH [$\chi^2(4) = 139.4, p < .001$]; LL[$\chi^2(4) = 200.3, p < .001$]; LH [$\chi^2(4) = 181.9, p < .001$]. To follow up on this finding, we compared all participant groups using a pairwise comparison with adjusted p-values, with results shown in Table 3. This means the fixation duration was the same for these pairs, either equally long or short. Both SL and LL show the same number and pairs of non-significant differences between AoIs, except for *PS-OP* in LL and *Oper-PS* in SL.

As for the SL group, non-significant differences were found between the *Var-DT* and *DT-Oper*, which suggests that participants

Table 3: Pairwise comparison of the fixation duration on AoIs with adjusted p-values (p) and effect size (r) for all participant groups. Effect size: SL = small (1), medium (1), large (8); SH = small (1), medium (0), large (9); LL = small (0), medium (2), large (8); LH = small (0), medium (0), large (10). Most pairs (bold) are significant, $p < .001$.

	SL		SH		LL		LH	
	p	r	p	r	p	r	p	r
Var-DT	.075	0.57	**.000**	0.74	.917	0.42	**.000**	0.80
Var-Oper	**.000**	1.03	**.000**	1.01	**.000**	1.12	**.000**	1.49
Var-PS	**.000**	1.18	**.000**	1.54	**.000**	2.41	**.000**	2.06
Var-Oper	**.000**	2.30	**.000**	2.80	**.000**	2.90	**.000**	2.92
DT-Oper	.075	0.46	1.00	0.27	.052	0.70	**.000**	0.69
DT-PS	**.000**	0.61	**.000**	0.79	**.000**	1.99	**.000**	1.27
DT-OP	**.000**	1.73	**.000**	2.06	**.000**	2.48	**.000**	2.12
Oper-PS	1.00	0.15	.374	0.52	**.000**	1.29	.149	0.57
Oper-OP	**.000**	1.27	**.000**	1.79	**.000**	1.78	**.000**	1.43
PS-OP	**.000**	1.12	**.000**	1.27	.512	0.49	**.000**	0.85

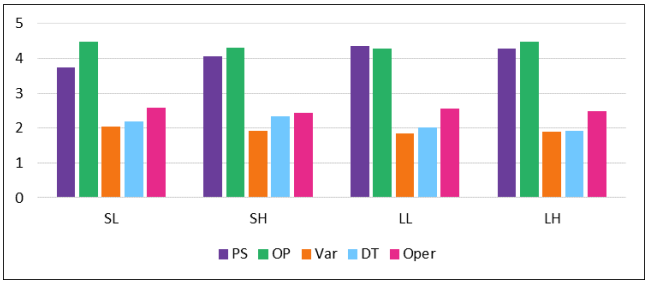

Figure 3: Fixation count for all participant groups on the five main AoIs. PS and OP showed the highest number of fixations, while Var has the lowest number of fixations.
Legend: short&low (SL); short&high (SH); long&low (LL); long&high (LH); problem statement (PS); example output (OP); variable(s) (Var); data type(s) (DT); operation(s) (Oper).

Table 4: Pairwise comparison of the fixation count on AoIs with adjusted p-values (p) and effect size (r) for all participant groups. Effect sizes: SL = small (1), medium (1), large (8); SH = small (2), medium (2), large (6); LL = small (2), medium (0), large (8); LH = small (2), medium (0), large (8). Most pairs (bold) are significant, $p < .001$.

	SL		SH		LL		LH	
	p	r	p	r	p	r	p	r
Var-DT	1.00	0.13	1.00	0.41	1.00	0.17	1.00	0.04
Var-Oper	1.52	0.52	.512	0.49	.066	0.68	.171	0.56
Var-PS	**.000**	1.62	**.000**	2.04	**.000**	2.30	**.000**	2.26
Var-Oper	**.000**	2.33	**.000**	2.27	**.000**	2.39	**.000**	2.44
DT-Oper	.687	0.39	1.00	0.08	.424	0.51	.253	0.53
DT-Ps	**.000**	1.49	**.000**	1.63	**.000**	2.13	**.000**	2.23
DT-OP	**.000**	2.20	**.000**	1.86	**.000**	2.21	**.000**	2.41
Oper-PS	**.000**	1.11	**.000**	1.55	**.000**	1.62	**.000**	1.70
Oper-OP	**.000**	1.81	**.000**	1.78	**.000**	1.71	**.000**	1.88
PS-OP	**.000**	0.70	1.00	0.22	1.00	0.09	1.00	0.18

were spending an equally short or long time on each pair. This further indicates that there were no differences of fixation duration spent on these AoIs. However, the significant difference between *PS-OP* suggests that the SL group had a longer fixation duration on these two AoIs, signifying that participants showed effort trying to understand the programming problems by focusing on *PS* or *OP*.

In the SH group, non-significant results were found for the two answer pairs *DT-Oper* and *Oper-PS*, which suggests that most likely the effective participants fixated longer on the *Oper*, *DT*, and *PS* across all of the problems posed. This is because these AoIs require substantial amount of thinking and potentially frequent validations between comprehension and problem solving processes.

Inspecting the LL group, non-significant differences were found for *PS-OP*, which suggests that the less effective participants most likely spent the same amount of time on requirements gathering in the *PS* and *OP* AoIs. Further, non-significant differences were found for *Var-DT* and *DT-Oper* potentially implying the same reason as described for the SL. It is also likely that the LL group shows a less organized problem solving strategy indicated by significant differences for the *PS-Var*, *PS-DT*, and *PS-Oper*. This could imply that the LL group could have frequented the *PS* while answering the *Var*, *DT*, and *Oper* questions.

The LH group only has one non-significant difference between the *Oper-PS* pair. The same amount of time spent on these AoIs implies that these were considered equally important in the assessment of the programming problems. Given that the LH group consisted of participants who had a long completion time and scored high, this finding could further infer that the time was spent on the *Oper* and *PS* AoIs to support the validation processes that typically occurs during problem assessment.

4.3.2 Fixation count. As indicated by Sharafi et al. [2015], eye tracking studies in software engineering considers the fixation count as an indicator to identify AoIs, which attract more visual attention. It is also used to report the amount of visual effort required to perform a task. For example, Goldberg and Kotval [1999] reported that a higher number of fixations spent on a stimulus reflects an inefficient approach to find information. In this study, the fixation count indicates the visual effort participants spent on the different AoIs. Figure 3 shows the average fixation count on the five AoIs for all participant groups: *PS* and *OP* received the highest number of fixations, while *Var* has the lowest number of fixations. The non-parametric test (Friedman's ANOVA) confirmed that the fixation count significantly changes across all AoIs for all participant groups: SL $[\chi^2(4) = 181.8, p < .001]$; SH $[\chi^2(4) = 138.2, p < .001]$; LL $[\chi^2(4) = 171.4, p < .001]$; LH $[\chi^2(4) = 209.1, p < .001]$. A closer inspection on the pairwise comparisons with adjusted p-values in Table 4 shows that the fixation counts are almost the same between *Var-DT*, *Var-Oper*, and *DT-Oper* for all participant groups as indicated by non-significant differences between these pairs. Additionally, the number of fixations are the same between *PS-OP* for the high groups and long groups (SH, LL, LH), suggesting that these two AoIs are treated as equally important in terms of the amount of attention. As could be expected, the short duration of the low group (SL) indicates that they spent a different amount of visual effort between *PS* and *OP*, suggesting that one of these AoIs could have received more mental processing amount.

4.4 Visual Analysis

We use scarf plots [Richardson and Dale 2005] for our AoI sequence analysis, which enables a closer assessment of the participants' reading behavior compared to more common eye tracking visualizations such as heat maps or gaze plots. Scarf plots better illustrate the visual distribution of gaze patterns on the identified AoIs. This produces improved insights about the participants' strategies in gathering requirements and solving the programming problems.

The following sections describe qualitative observations of the AoI sequence analysis, by first going through the main patterns observed across all participant groups. Then we highlight the most noticeable patterns characterizing the specific participant groups which is discussed based on their comprehension strategy, sequence of providing solutions in the answer AoIs, frequency of revisits, and interaction with other AoIs.

4.4.1 General Pattern. In terms of requirements gathering, generally, the participants were found to read the problem statement at the beginning of their assessment to understand the programming problem posed. This initial reading of the problem statement varied in duration. Figure 4 shows long strips of purple, which corresponds to *PS*, for most of the participants, especially at the beginning of the task. This supports our initial findings suggesting that participants followed the top-down nature of the stimulus, in which they needed to understand the programming problem prior to answering the questions. There were also frequent visits to *PS* as participants began to attend the answers.

The statistical analysis showed that time spent on *OP* (Figure 4, green) for all participant groups were the longest of all AoIs. The scarf plot further reveals that this AoI was visited frequently in chunks especially when participants tried to identify the number of variables (Figure 4, orange) that needed to be defined for the programming problem. Although there were multiple revisits to *Var*, participants only spent a short time during each visit. Hence, this AoI produced the shortest total fixation duration when participants were deciding on the number of variables. Determining the most suitable operation(s) for the programming problems took the longest fixation duration among the three answer AoIs (*Var*, *DT*, *Oper*). All of these findings confirm the reported statistical analyses.

4.4.2 Group Specific Patterns. We could not find consistent patterns unique for each participant group (SL, SH, LL, LH). Therefore, we discuss more general patterns found or shared between some of them. There were two main strategies participants adopted in gathering requirements or understanding the programming problems. We found that the short groups (SL, SH) spent time to understand the nature of each programming problem by reading the problem statement and example output individually, making sure they understood the content (either *PS* (purple) or *OP* (green) before moving to the next part. This behavior can be seen in Figure 4 on the left for both of the SL and SH groups. On the contrary, the long groups (LL, LH) were more comfortable to switch between *PS* (purple) and *OP* (green) in the process of understanding the programming problems, which can be seen in Figure 4 on the right. This could indicate that these participants' were trying to confirm what they understood from the problem statement by inspecting the outcome shown in the example output.

More specifically, the LH group spent a shorter time on *PS* at the beginning for all of their problem assessments, while the short groups (SL, SH) had a short beginning for the medium and hard questions. In contrast, the LL group spent a longer time on *PS* at the beginning of their assessment. A possible reason is that these participants paid more attention to *PS* to understand the programming problem suggesting that terms used in the problem description are important to assist them in coming up with their answer. Given that *OP* is visited most often and has the longest fixation duration, this suggests that the long groups had more transitions between *OP-DT* and *OP-Oper* (cf. Figure 4, green (*OP*), blue (*DT*), and pink (*Oper*)). An opposite pattern was found for the short groups.

In terms of the order of answering the questions, the LH group commonly exhibited a strategy akin to the breadth-first search that traverses from top to bottom and left to right. This indicates a structured approach adopted by those who took a long time but reached correct answers. Another strategy adopted by participant groups was a mixed-variation of the order when attempting to answer the questions. In evaluating the data types question, it appears that the short groups (SL, SH) answered this section in one attempt without showing frequent revisits, indicating focused attention in dealing with this type of question. Instead, the long groups showed a tendency for frequent revisits while identifying the data types. Further analysis on how participants evaluated the operation question showed that the short groups had a tendency to complete this question at once. This pattern is reflected by multiple revisits, often seen as a switching between *PS* (purple) and *Oper* (pink), among the long groups. However, the short groups showed less frequent revisits to *PS*.

5 DISCUSSION

Research Question 1. How do participants comprehend and solve programming problems?

We designed the questions in the study in a way to allow us to determine the procedural knowledge of participants for solving programming problems. Stolinska and Andrzejewska [2017] identified procedural knowledge as participants' behavior that reflects their ability to solve algorithmic tasks via eye tracking studies. In their study, the authors identified the state of participants' procedural knowledge by the percentage of scores they achieved from solving flowchart problems. However, limited details were given about the implication of identifying procedural knowledge on cognitive processes in solving programming problems.

In our study, the programming problems posed, required participants to first analyze its requirements, then identify solutions in terms of the number of variables and its corresponding data types followed by the types of operations that can be used to solve the problem. After understanding the problem statements, participants are flexible to answer in any preferred order as they solve the problem. We view this activity as an opportunity to gain better insights about their procedural knowledge or determining how the problems are solved. Our findings showed that all participants inspected the problems first in which the output area received more attention than the textual problem description. This implies a lot of participants' effort was allocated on the example output to understand the

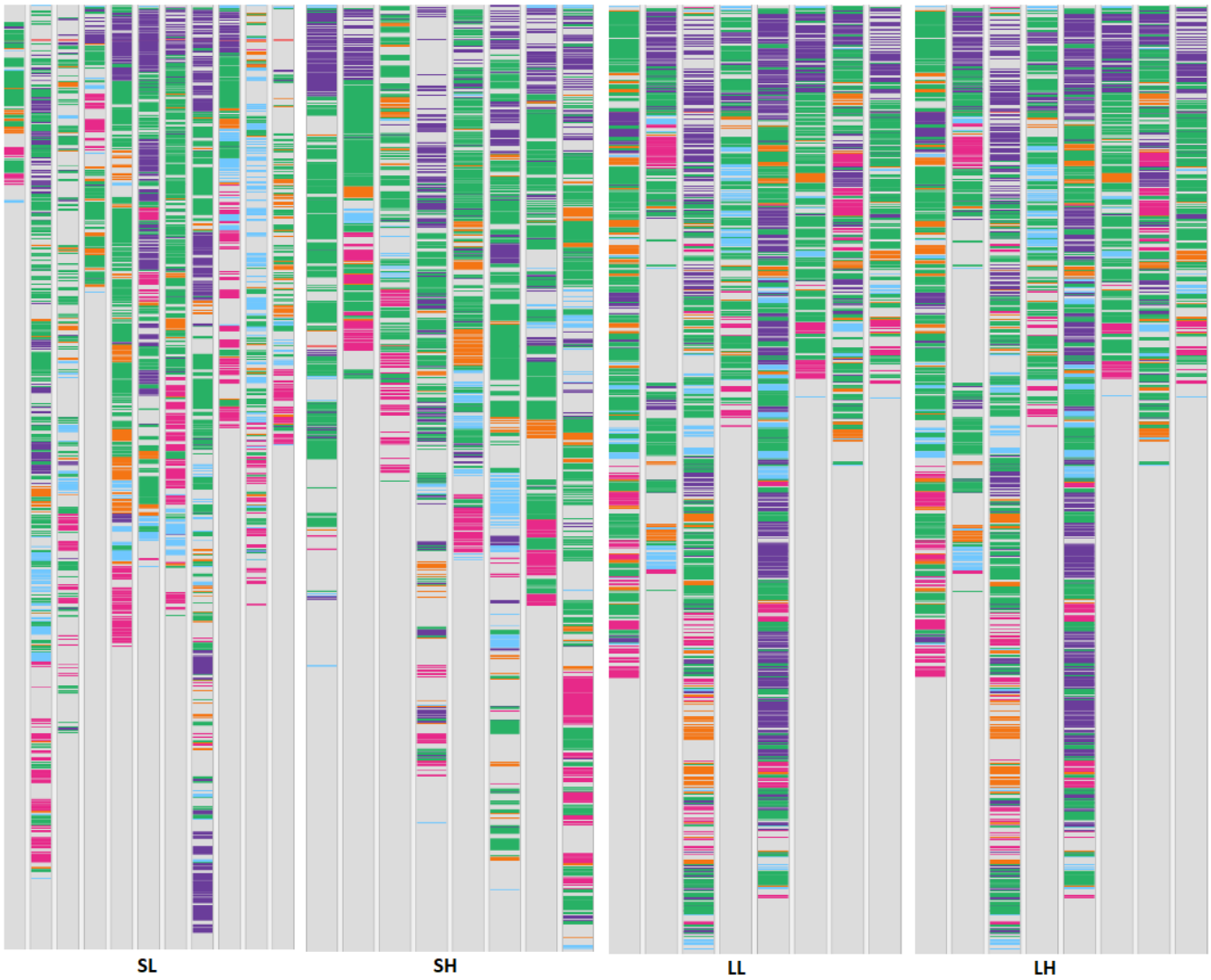

Figure 4: Scarf plot of all participants by participant group for one stimulus.
Legend: short&low (SL); short&high (SH); long&low (LL); long&high (LH). Colors: purple = problem statement; green = example output; orange = variable(s); blue = data type(s); pink = operation(s). All visualizations have been normalized to the same height.

problem. The longest duration spent on the example output area compared to other AoI probably indicates that participants were trying to trace a match or find an analogy between the problems via its outcome with their mental schema. The relevant retrieved schemas provided assistance in interpreting the new information by giving access to the relevant prior knowledge. In other words, some specific areas of the example output could have provided hints that determine the organization of their subsequent solution strategies.

Further inspection on the common sequences participants used when solving programming problems of this format, collectively indicated that participants first identified the data types, then the variables, followed by the operations. The identification of data types first can be viewed as a process requiring the least effort that takes less cognitively demanding mental strength. Conversely, determining operations is considered as the most cognitive demanding task requiring logical processing and abstract thinking and was left to the end of the task. However, additional investigation to identify whether this pattern persists without the availability of the example output with random order of the answer parts is worthwhile, to confirm the state of procedural knowledge of participants when solving algorithmic problems of this kind. Identifying the solution as a paper-based task could also provide useful insights.

Research Question 2. Which classification of reading strategies for programming problems can we derive?

In our analysis, we acquired four groups identified as SL, SH, LL and LH, which differ based on short and long completion times as well as based on high and low scores.

Previous studies have indicated that retrieval of mental schemas helps to solve problems [Détienne and Soloway 1990; Rist 1989].

This is most effective when the context or problem under investigation fits the retrieved mental schema. We argue that the high performing participants (SH, LH) encompass a comprehensive schema network consisting of a large number of different schemas for solving programming tasks acquired from previous knowledge of past experience. This could also be viewed as having a good amount of declarative knowledge. It could be that these participants (33%) have greater expertise in programming as confirmed by a few years of programming education, familiarity with programming problems, and the good score in their *Fundamental of Computer Programming* course as indicated in the post study questionnaire.

In terms of the amount of time spent to understand the problem between the high performing groups, the SH participants took more time to evaluate the problem statement and the example output, while the LH participants spent less time on this activity. The differences in how these groups assessed the requirements of the problem indicate that participants who had a short completion time could have effectively used their time to evoke the relevant schemas while analyzing the problem requirements. In comparison to the LH and SL groups, shorter time spent on the solution section for this group (SH) could further reveal that once the relevant schemas are identified, the process of identifying the solutions in terms of variables, data types, and operations become relatively easy, increasing the possibility of the problem being identified as well-defined by the participants. Hence, requiring less demanding cognitive effort. This is further supported by the fewer number of transitions between the different solution areas enabling each part of the solution to be attended individually, one at a time. However, the same number of fixations across all groups could imply that participants spent an equal amount of visual effort on these areas.

A comparison between the lower performing participants (SL, LL) leads us to suggest that these groups, potentially classified as novices compared to the higher performing group, hold an incomprehensive schema network or limited number of effective schemas for programming problems. These participants could have tried to find a matching mental schema for a specific problem. The limited number of connections associated with other related schemas could be due to an insufficient amount of declarative or conceptual knowledge possessed by these groups of participants. Therefore, upon assessing the problems presented, probably considered as new problems encountered, these groups needed to set up a new schema by integrating their limited available knowledge from their existing schemas for the given task. This is supported by the higher number of transitions during the problem comprehension (between problem statement and example output) and frequent visits between the solution areas. These groups attempted multiple and frequent validations, which eventually lead to an ineffective solution. No differences between the solution areas for both low performing groups unveiled a less organized strategy in identifying solutions.

To understand the problem requirement, the SL participants used the same strategy as the SH group, while the strategy of the LL group is similar to the LH group. In the case of the SL group, shorter time could have been invested to identify the problem requirements. The non-significant difference between *PS-OP* for the LL group suggests that this group, probably, could not benefit from the hints and terms provided in these areas to assist their problem comprehension. However, a different amount of time switching between these areas for the SL group may suggest that participants spending shorter time could have attempted some level of comprehension, although this may not have been effective enough. Furthermore, both the SL and LL groups spent relatively short amounts of time to identify the operations for the programming problems and almost equal amounts of time to identify the types of variables. This implies that limited logical reasoning and abstraction thinking occurred among the low performing participants.

In terms of problem solving strategies that relate to expertise, we have only been able to characterize the general strategy adopted by all participants as a top-down approach given the format that represents the programming problems. We have not been able to specify problem solving strategies for each participant group with the current data. Examples of these strategies are divide-and-conquer, hill climbing, and trial-and-error. The identification of these strategies is valuable as the outcome could enable educators to provide meaningful feedback that promotes reflection and support, which improves the way participants learn and write computer programs. A limitation of the present stimuli design relates to the order of areas (Var, DT, Oper) and answer options within each area. It may be that over time participants became familiar with the stimulus layout and did not pay much attention to the choices given. Therefore, appropriate stimulus design could serve as a good starting point for further work along this goal.

The last question of each stimulus asked participants' opinion about the level of difficulty for the problem. The majority of participants rated the algorithmic problems as easy indicating that their problem comprehensibility was good. Their opinion to some extent is consistent with the total scores because all participants reached an accuracy above 50%. Seventeen of the participants scored above 75%. However, when compared to the classified grouping based on their effort spent, the scores may not be representative of the reading patterns. Therefore, it could be that the stimuli were not representing the right level of difficulty for these participants.

6 CONCLUSION AND FUTURE WORK

In this paper, we reported on the findings of an eye tracking study conducted with undergraduate computer science students who completed a series of problem solving tasks of programming problems. Our main goal was to analyze participants reading behavior and patterns of problem solving strategies using a visual AoI sequence analysis based on the students' performance and visual effort. Findings from this study indicate to teachers that participants regardless of their performance level, read, evaluate, and solve programming problems differently. There is a need to better understand students' reading and problem solving behavior in this respect to provide improved guidance and effective feedback. Our results indicate that the activation of relevant prior knowledge is beneficial in solving programming problems. Therefore, retrieval practice and elaboration on their understanding of the programming problems that serves as supporting strategies in problem solving are recommended in teaching and learning activities. This is because connecting new materials to activated prior knowledge potentially enables participants to create a more organized and rich information for the new problem under investigation.

ACKNOWLEDGMENTS

The authors would like to the students who participated in the study. The authors appreciate Dr Mahmoud Danaee's help in providing guidance on statistical analysis and the reviewers' suggestions for revisions. This study is supported by Grant No: RP030-14AET and RP061B-18SBS.

REFERENCES

Tanja Blascheck, Kuno Kurzhals, Michael Raschke, Michael Burch, Daniel Weiskopf, and Thomas Ertl. 2017. Visualization of Eye Tracking Data: A Taxonomy and Survey. *Computer Graphics Forum* 36, 8 (2017), 260–284.

Martha Crosby and Jan Stelovsky. 1990. How do we Read Algorithms? A Case Study. *Computer* 23, 1 (1990), 25–35.

Françoise Détienne and Elliot Soloway. 1990. An Empirically-derived Control Structure for the Process of Program Understanding. *International Journal of Man-Machine Studies* 33, 3 (1990), 323–342.

Rodrigo Duran, Juha Sorva, and Sofia Leite. 2018. Towards an Analysis of Program Complexity From a Cognitive Perspective. In *Proceedings of the ACM Conference on International Computing Education Research*. ACM, 21–30.

Joseph Goldberg and Xerxes Kotval. 1999. Computer Interface Evaluation using Eye Movements: Methods and Constructs. *International Journal of Industrial Ergonomics* 24, 6 (1999), 631–645.

Anabela Gomes and António José Mendes. 2007. Learning to Program – Difficulties and Solutions. In *International Conference on Engineering Education*. ACM, 1–5.

Kenneth Holmqvist, Jana Holsanova, Mari Barthelson, and Daniel Lundqvist. 2003. Reading or Scanning? A Study of Newspaper and Net Paper Reading. In *The Mind's Eye*. Elsevier, 657–670.

Lai Meng Lung, Tsai Meng Jung, Ying Yang Fang, Yuan Hsu Chung, Chien Liu Tzu, Wen Yu Lee Silvia, Hsien Lee Min, Li Chiou Guo, Chong Liang Jyh, and Chung Tsai Chin. 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 Cheng. 2018. A Survey on the Usage of Eye-tracking in Computer Programming. *Comput. Surveys* 51, 1 (2018), 5.

Daniel Richardson and Rick Dale. 2005. Looking To Understand: The Coupling Between Speakers' and Listeners' Eye Movements and Its Relationship to Discourse Comprehension. *Cognitive Science* 29, 6 (2005), 1045–1060.

Robert Rist. 1989. Schema Creation in Programming. *Cognitive Science* 13, 3 (1989), 389–414.

Dario Salvucci and Joseph Goldberg. 2000. Identifying Fixations and Saccades in Eye-tracking Protocols. In *Proceedings of the Symposium on Eye Tracking Research & Applications*. ACM, 71–78.

Carsten Schulte, Tony Clear, Ahmad Taherkhani, Teresa Busjahn, and James Paterson. 2010. An Introduction to Program Comprehension for Computer Science Educators. In *Proceedings of the ITiCSE Working Group Reports*. ACM, 65–86.

Zohreh Sharafi, Timothy Shaffer, Bonita Sharif, and Yann-Gaël Guéhéneuc. 2015. Eye-Tracking Metrics in Software Engineering. In *Asia-Pacific Software Engineering Conference*. IEEE Computer Society Press, 96–103.

Kshitij Sharma, Katerina Mangaroska, Halvard Trætteberg, Serena Lee-Cultura, and Michail Giannakos. 2018. Evidence for Programming Strategies in University Coding Exercises. In *European Conference on Technology Enhanced Learning*. Springer, 326–339.

Elliot Soloway. 1986. Learning to Program = Learning to Construct Mechanisms and Explanations. *Commun. ACM* 29, 9 (1986), 850–858.

Elliot Soloway, Jeffrey Bonar, and Kate Ehrlich. 1983. Cognitive Strategies and Looping Constructs: An Empirical Study. *Commun. ACM* 26, 11 (1983), 853–860.

Elliot Soloway and Kate Ehrlich. 1984. Empirical Studies of Programming Knowledge. *IEEE Transactions on Software Engineering* SE-10, 5 (1984), 595–609.

James Spohrer, Elliot Soloway, and Edgar Pope. 1985. A Goal/Plan Analysis of Buggy Pascal Programs. *Human–Computer Interaction* 1, 2 (1985), 163–207.

Anna Stolińska and Magdalena Andrzejewska. 2017. Analysis of the Strategy Used to Solve Algorithmic Problem: A Case Study Based on Eye Tracking Research. In *New Trends in Analysis and Interdisciplinary Applications*. Springer, 77–86.