TECHNISCHE UNIVERSITÄT
CHEMNITZ

Program Comprehension in Numbers and Words Algorithms

Master Thesis

by

Anirudh Adavalli

at

Chemnitz University of Technology
Department of Computer Science

Course of Studies
WEB ENGINEERING

Professor: Dr. Janet Siegmund
Chemnitz University of Technology

Supervisor: MSc. Arooba Aqeel
Chemnitz University of Technology

Chemnitz, April 2021

Contact details: Anirudh Adavalli
[redacted]

Dr. Janet Siegmund
Professorship of Software Engineering
Chemnitz University of Technology
09111 Chemnitz
janet.siegmund@informatik.tu-chemnitz.de

Arooba Aqeel
Straße der Nationen 62
Chemnitz University of Technology
09111 Chemnitz
arooba.aqeel@informatik.tu-chemnitz.de

Declaration of Authorship

I hereby declare that this master thesis was independently composed and authored by myself.

All content and ideas drawn directly or indirectly from external sources are indicated as such. All sources and materials that have been used are referred to in this thesis.

The thesis has not been submitted to any other examining body and has not been published.

_____

Place, date

_____

Signed:Anirudh Adavalli

Abstract

Program comprehension is about how programmers comprehend the source code. Since the past three decades, program comprehension research has yielded a plethora of theories and methods that have aided in the interpretation and explanation of how programmers comprehend source code. A better understanding of programmers aids in developing efficient tools and methods to help the programmer in comprehension tasks and also teach programming effectively. Various measuring methods, instruments, and technology have been used to investigate various aspects of program comprehension. Technologies such as fMRI, Eye-tracking, EEG, etc were used to understand cognitive processes associated with comprehension. In this thesis work, the difference in program comprehension of numbers and words algorithms is explored. The correctness and response time are measured for numbers and words algorithms to observe the differences. The study is carried out using an online questionnaire platform called SoSci Survey and to observe the difference in approaches for numbers and words algorithms a think-aloud study is employed. An online meeting is conducted with the participants where the participants are expected to verbalize their thoughts while answering the questionnaire. The responses are collected in the same survey platform. The transcripts are generated from the audio of the participants. The correctness and response time for number and words algorithms are formulated from the survey data. Transcripts are analyzed to gain more insights. The study revealed that there is no difference in the correctness and response time for identical numbers and words algorithms.

Keywords: Program Comprehension, Numbers and words algorithms, Correctness and response time, Think-aloud protocol.

Acknowledgements

The successful outcome of this work is not possible without valuable guidance and assistance from many people who deserve a token of acknowledgment right from my heart. I would like to thank my supervisor Arooba Aqeel, and my professor, Dr. Janet Siegmund, for providing me this opportunity and for their support and advice during the thesis research. I would also like to thank Chemnitz University of Technology for providing the resources. Lastly, I would like to thank everyone who participated in the research and provided valuable feedback.

Contents

List of Figures

List of Abbreviations

BOLD	**B**lood **O**xygenation **L**evel **D**ependent
DV	**D**ependent **V**ariable
EEG	**E**lectroencephalogram
fMRI	**f**unctional **M**agnetic **R**esonance **I**maging
fNIRS	**f**unctional **N**ear - **I**nfrared **S**pectroscopy
FIFO	**F**irst **I**n **F**irst **O**ut
IV	**I**ndependent **V**ariable
LRU	**L**east **R**ecently **U**sed
MRI	**M**agnetic **R**esonance **I**maging
MV	**M**itigating **V**ariable
OBS	**O**pen **B**roadcaster **S**oftware

Chapter 1

Introduction

Program comprehension or code comprehension describes the process of how developers comprehend source code, it is a vital human element in software engineering. Program comprehension is the primary activity of software developers. It is an important cognitive process in software development as developers spend most of their time understanding source code [1].

A better understanding of the cognitive process involved in programming would help in determining better ways of teaching programming, understanding what teaching practices would affect program comprehension of programmers. Also, it helps in the design and development of efficient programming languages and software tools that aid the developers in their day-to-day lives and supports them in writing better software [2].

The study on program comprehension is been carried out for 30 years. In the earlier days of research on program comprehension, researchers used various approaches like Memorization, Think-Aloud Protocol, and Comprehension tasks to measure program comprehension. These studies led to the program comprehension models such as Top-Down Comprehension, Bottom-Up Comprehension, and Integrated models [3].

Since the 1990s, researchers have used techniques like neuro-imaging and eye-tracking to gain more insights into programmers brains. Siegmund, Janet, et al used fMRI for the first time to study program comprehension where they found that Brodmann areas 6, 12, 40, 44, and 47 of brain are activated while program comprehension. Other

researchers have also used near-infrared spectroscopy and electroencephalography (EEG) to study program comprehension [4] [3].

1.1 Problem Statement and Motivation

With over 3 decades of research on program comprehension, a number of theories, models, and tools have emerged which describe how programmers think while they are trying to understand source code. To understand program comprehension better insights into the underlying cognitive process have to be obtained. Even with a significant amount of research, little is known about the cognitive process associated with program comprehension. Thus it is becoming difficult for researchers to suggest appropriate coding conventions, tools, and programming languages to aid developers in their day-to-day work. If the programming tasks and the amount of cognitive effort can be calculated and linked, it could be possible to determine which types of problem solving, activities, and code segments are difficult for programmers. Such identification would help in improving training methods, education and developing programming languages and tools to for programmers [3] [2] [4] [5].

Considering the theories models and tools that have evolved over the period of time in program comprehension, the motivation behind this study is to understand the behavior of programmers for numbers and words in programming. This study will help in understanding if the programmers perceive the numbers and words differently in programming. The differences in the amount of effort and the difference in approach for tackling the number and words can be observed from the results of this study. The differences in the correctness of responses and response time for numbers and words are the two factors considered to be observed in this study. The design of the study includes an experiment that needs to be conducted with participants who have a computer science background. The study focus on understanding the difference in the behavior of the programmers for number and words, while the programmers solving small code snippets. The problem statement this study address is whether

there would any differences in programmers behavior for numbers and words. The study was initially intended to be conducted with the help of a web-based eye-tracking technology but another approach "think-aloud protocol" had to be used to observe the participants approach towards numbers and words. Due to the pandemic and restrictions on human contact, the experiment is designed to be conducted online. An online survey tool called "SoSci Survey" is used to conduct the experiment.

1.2 Objective

The objective of this thesis is to study the behavioral differences in programmers while they are comprehending the numbers and words algorithms.

Based on the above objective the following two research questions are addressed in this thesis work.

1. **RQ1:** Does the correctness of programmers differ for numbers and words algorithms?
2. **RQ2:** Does the response time of programmers differ numbers and words algorithms?

Initially, the study is intended to be conducted with an eye tracking; however, a think aloud protocol is used to observe the differences in the subjects approach to tackle the tasks instead of eye tracking.

1.3 Context

An online survey will be conducted with 12 participants from the field of computer science. Participants will be asked to take an online survey. Individual participants will be invited to an online meeting. Participants are given a link to the survey, which consists of questionnaires on comprehension tasks to be answered while verbalizing their thoughts. Their voices and screen were recorded during the meeting session using

a screen recorder so that transcripts can be produced. These transcripts and responses will be analyzed to generate meaningful data that will help prove the hypothesis.

1.4 Thesis Structure

This section highlights the contents of each chapter in this thesis report. The figure 1.1 presents the chapters and sub-sections. The chapter wise outline is as following:

- **Chapter 1 (Introduction):** This chapter introduces the thesis work. Details regarding the motivation, problem statement, context, and thesis structure are provided in this chapter
- **Chapter 2 (Literature Review):** This chapter will provide a review of past relevant research to this thesis work. It shares the details of program comprehension, comprehension modes, and approaches used in measuring program comprehension.
- **Chapter 3 (Experiment Planning):** This chapter provides detailed planning and design of the experiment it included a detailed description of goals, participants, experimental material preparation, tasks to be performed by the participants, and data analysis procedure.
- **Chapter 4 (Conduct):** This chapter provides insights into the data collection process, the procedure to experiment.
- **Chapter 5 (Data Analysis and Results):** This chapter highlights the data analysis procedure and interprets the results.
- **Chapter 6 (Discussion):** This chapter provides an overview of the results and provided a discussion on important aspects of the study.
- **Chapter 7 (Conclusion and Future Work):** This chapter provides the conclusion to this research and the future scope of the research topic.

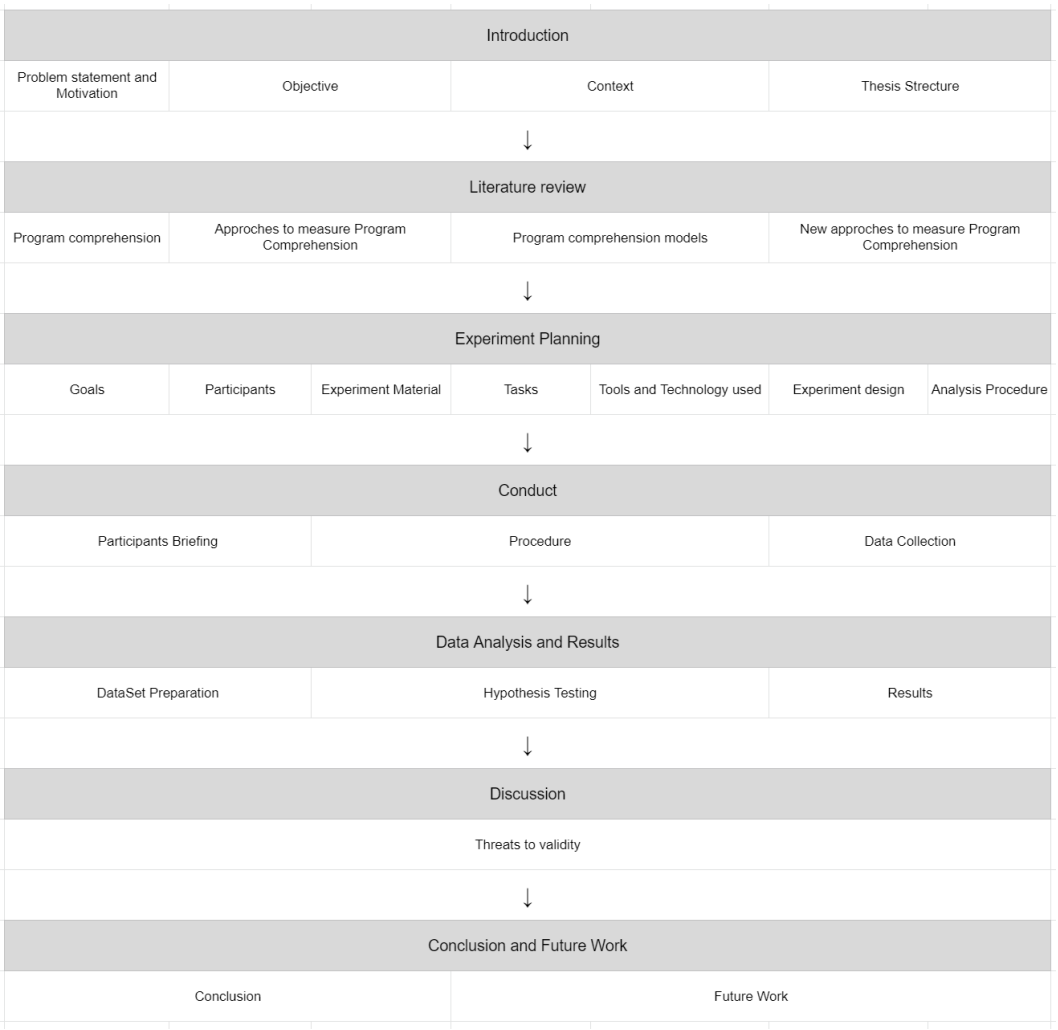


FIGURE 1.1: Thesis Structure

Chapter 2

Literature Review

This section highlights the past research in code comprehension, comprehension models and approaches used to measuring code comprehension.

2.1 Program Comprehension

Program comprehension is a study that explores how programmers understand programs. One can comprehend a program if he/she understands its behavior and structure. Understanding the cognitive processes of developers, such as the processes by which they comprehend programs, is a prerequisite for building tools, environments, and methods that support software development in an appropriate manner [6] [7] [8].

Program comprehension is an important activity in the software development life cycle. In software maintenance activities, programmers spend almost 60 percent of their time comprehending the source code and thus program comprehension will have a notable impact on the cost of software maintenance. If a better understanding of program comprehension is gained it would help in developing tools and technologies that simplify program comprehension thereby reducing time and cost. To make program comprehension an easy task, it has to be understood and measured. However, program comprehension involves a complex cognitive process that can only be measured with the help of controlled experiments [9]. There are numerous approaches to measure program comprehension, few of them used in the past and present are presented in the sections 2.2 and 2.4.

2.2 Approaches to measure Program Comprehension

2.2.1 Recall

Free recall of source code was another way to assess software comprehension. Allowing developers to remember source code to see if they understood it might seem strange today, but memorization and eventual recall of source code is an important part of the development process [10]. In general, recall tests include giving a participant a fragment of code and asking them to research it for a fixed period of time. The code is then removed/hidden and the subjects are asked to recall as much of the code as they can. Both of these steps are repeated several times in some cases [11].

Shneiderman, who wrote many popular reports on program comprehension at the time, compared programmers' abilities to musicians' ability to recall every note in music tracks or symphonies: He recommended that programmers develop the very same ability to comprehend entire programs in detail [12]. In one of the studies, Shneiderman examined two different patterns of a program, in one of the versions the programming statements were in an executable order and in the other version the statement pattern was fragmented [10] and he noticed that the one with the proper executable manner was easier to be memorised and this was also linked to the programming knowledge (the participants are good at memorising the code with the one which is in executable order).

Small code snippets were provided to the developers by Soloway and Ehrlich [13]. The goal of the task was to recall the code snippets in exactly the same way it was provided to the programmers. The researchers evaluated the results and noticed that the professional programmers were more dependent on the coding standards such as the meaningful variable names, that when these coding standards are broken, they are as sluggish as beginners. Recalling method was also used by Pennington to test program comprehension and studied the influences caused by priming on response time [11]. Priming is the process of responding quickly to the target stimulus if

you have seen similar stimulus before. As two stimuli are saved nearly together in participants memory, the target stimulus gets activated by the similar stimulus and allows participants to respond quickly. During the experiment, the participants were asked to determine if the code of lines provided in a sequential order were included in the code snippets they were looking at. If the code of line was followed by the similar statement then the response time was quicker [3] [14].

2.2.2 Comprehension Tasks

Another activity for developers in Soloway and Ehrlich's analysis was to "fill in the blanks," or fill in a left-out part of the code to complete the code snippet. Programmers can only provide the missing part of the code accurately if they already indeed understand it [35]. Boysen allowed Programmers to decide between easy expressions, such as $x < 5$, were true or false, or find the value of a variable for a more complex code of lines, such as if $x < 5$ then $y = 1$ else $y = 2$ [2]. Programmers' correctness and response time were calculated by Boysen. This challenge is more straightforward than asking Programmers to recall code because it specifically asks them to comprehend source code. He discovered that different expressions and operators result in various response times, and that true statements are executed more quickly [3]. Comprehension tasks generally entails giving participants a piece of the program that is missing a segment. The code snippet showed had never been seen before by the experiment participants, who were expected to complete a blank segment in the code snippet. Instructions or explanations of how to use the program were not provided to the Participants [14].

2.2.3 Think-Aloud Protocol

In the past various methods related to self-reporting were used to measure cognitive processes. Self-reporting methods have been used in cognitive processes observation for over a hundred years [15]. In this self-reporting method, subjects are asked to

verbalize their experiences, how they are feeling and what they are thinking while they are doing certain tasks. The subjects are audio or videotaped, these taps are used to generate transcripts and the transcripts are analyzed. Think-aloud is one such method. The think-aloud method was traditionally used in psychology for research on cognitive processes [16].

Since then think-aloud has been used in various disciplines. It has been used to study decision-making in the medical domain. Other domains used in this method are education and usability studies where researchers studied learning environments. In software engineering think-aloud is used to study program comprehension, debugging and differences in novices and experts [17] [18]. For example, Shaft and Vessey in their research used the think-aloud method to study developer's approaches for familiar and nonfamiliar programs. The results of the study showed that the programmer's employee hypothesis for familiar programs and they employee inferences for nonfamiliar programs [19]. Mayrhauser and Vans conducted a study with programmers who were working on a maintenance task. The programmer is instructed to verbalize their thoughts while working on maintenance tasks. The material used for this study is a software application with over forty thousand lines of code. The results of the study showed that programmers develop various models while understanding source code and they switch between different models [20].

In the Think-aloud method, there are two roles one is an observer and the other is subject. The observer takes note of whatever the subject is verbalizing. Data is collected from only one subject at a time. This method is expensive in terms of cost and effort. If the number of participants is high the cost and efforts to organize the study will raise. Usually, the number of subjects for the think-aloud method is very less in software engineering researches [21] [6].

Some of the other techniques used by researchers are:

- **Maintenance task:** A maintenance task on a chuck of code typically takes the form of an extension, elimination, or debug task. The completeness, correctness,

and time are taken to complete the assigned task are normally used to determine a programmer's level of comprehension. This technique has been used several times by researchers who want to learn more about a participant's cognitive processes [22].

- **Subjective rating:** A rating by participants is mainly the participant's self-evaluation of their interpretation of a code segment. Although the descriptive metric has been debated in some sections, it has been used to assess comprehension levels for the past two decades [23] [14].

- **Label/group code** Participants are asked to group and/or mark parts of code snippet in these experiments. Participants in Rist's experiment [24] clustered together increasingly larger and larger pieces of code snippet and produced details explaining why they did so [14].

- **Code coverage/optimisation** The experiment by Tapp and Kazman [25] was formulated to analyse the benefits of fonts and colors which help structuring the programs simpler to grasp it. There were two tasks, one was to use a unit test to make sure that all the lines of code in the program was executed (code coverage), and the other was to optimize a chunk of code that is low in performance while execution [14].

- **Call graph** While comparing typographic representations of code, Oman and Cook [26] used the Call graph technique. They gave their participants the task of completing an unfinished call graph for a segment of the X-Windows module written in the C programming language. Participants were evaluated based on how well they added additional nodes and edges to the call graph [14].

2.3 Program Comprehension Models

The approaches to measure program comprehension mentioned in section 2.2 led to the development of various models that explain the way programmers try to understand

source code [3] [5] [27]. Important program comprehension model are presented here:

2.3.1 Top-down Comprehension Model

Where the code or form of code is known, top-down interpretation shown in figure 2.1 is used. The program could then be broken down further into components that make up that device type. New programming can presumably be interpreted completely from the top down if indeed the programmer had previously learned code that did a similar thing and was designed in much the exact format.

Strategic, tactical, and execution plans are included in this model. Strategic plans define a program's or algorithm's overall strategy and determine activities that are not language-dependent. Local tactics for addressing the problem are tactical plans, and they include language-independent algorithm specifications. Language-dependent implementation plans have been used to carry out strategic planning which includes real code fragments. Top-down is used to construct a mental model. It has a set of goals and plans that are arranged in a hierarchical order. Decomposing objectives into plans, and plans into lower-level plans, is made easier with discourse rules and beacons. In most cases, shallow reasoning is used to link the structural modules. Figure 2.1 shows the three main components of the model. The triangles describe programming plans or discourse rules. For instance, a portion of the discourse rules would include the following:

- Variables updated the same way as initialized
- No unwanted code
- The condition must be potentially valid if there is a test for a condition
- Don't do double work with a program in an inconspicuous way
- When a statement is intended to execute only once then an If condition is considered; perhaps when the statement needs to be executed repeatedly a While condition can be used.

The diamond reflects the mechanism of comprehension. Internal or external program representations are shown by the rectangles. (Documents such as specifications or design documents; source-code; references, or maintenance manual; and other relevant documents are examples of external representations plans and schemes are examples of internal representations). By establishing objectives, the comprehension method matches external representations to programming strategies using principles of discourse to pick designs. The internal representation is modified to fit the newly gained information after a match is completed. Following that, the revised mental representations are saved as latest update. Comprehension starts with a high-level objective and then creates the detailed sub-goals that are needed to accomplish the high-level objective. Based on the current mental representation's emphasis, program documents and code evoke execution, strategic plans [28] [13] [29].

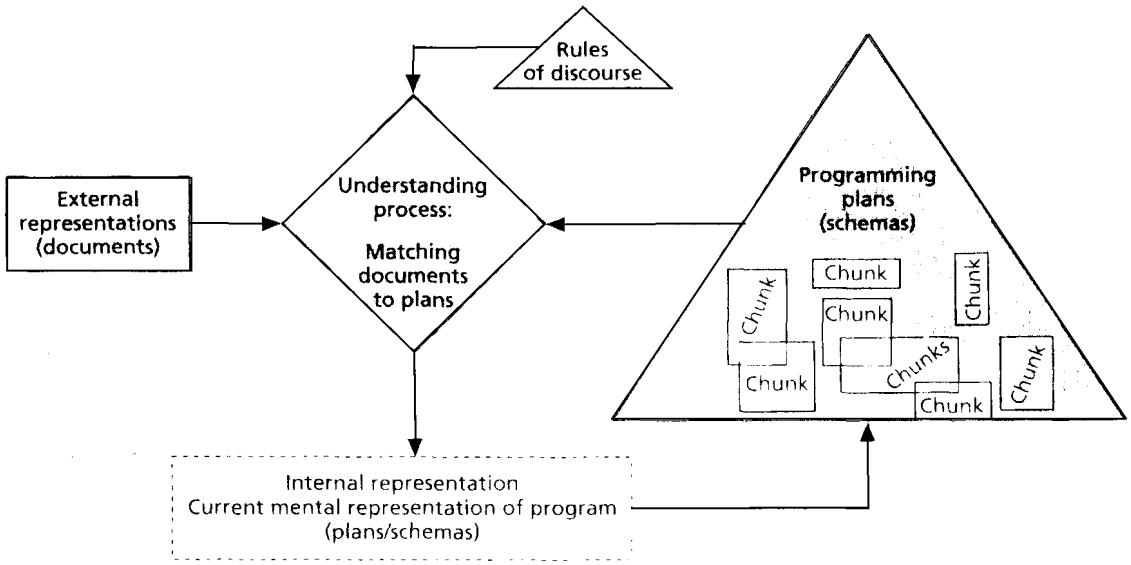


FIGURE 2.1: Top-down Comprehension Model [28]

.

2.3.2 Bottom-up Comprehension Model

A program model and a situation model are two conceptual representations that comprehension produces. Typically, the program model comes first, followed by the

situation model. Pennington discovered that when developers first encounter code, the first mental representation they create is a control-flow program framework known as the program model. This interpretation which is constructed from the bottom up comprehension model shown in figure 2.2 using observatories, defines the program's basic blocks of code control primes.

Pennington explains program model construction using sentence structures and programming plan information. Micro structures are chunked into macro structures and cross-referenced to construct the program model. During the comprehension task, programming plan information, which consists of programming principles, utilizes previous data and implies future initiatives for processing in long-term memory. Strategy knowledge from the operating - system environment includes least recently used (LRU) page substitution for memory management. Queue such as first-in first-out can be implemented with data structure information [28] [11].

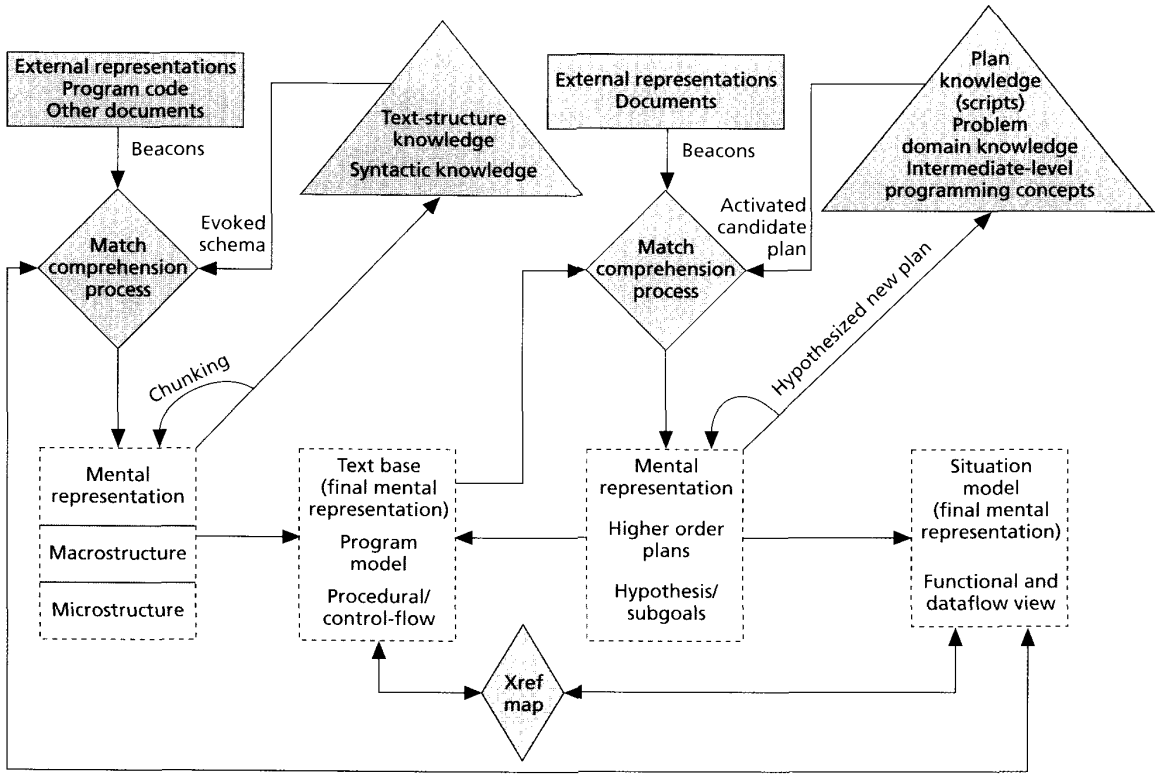


FIGURE 2.2: Bottom-up comprehension model [28]

.

2.3.3 Integrated Metamodel

The top-down, scenario, and software models, as well as the knowledge base, make up the integrated code comprehension model as shown in figure 2.3. The first three reflect comprehension processes. The very first three represent the stages of understanding. The fourth is required for the other three to be completely finished. Each element is included in the program's internal state as well as the method for constructing it. The knowledge base provides information about the comprehension task to the method and preserves any new or implied knowledge. When it comes to wide ecosystems, a range of approaches to comprehension is needed. As a result, the integrated model integrates Soloway, Adelson, and Ehrlich's top-down comprehension with Pennington's bottom-up explanation. Studies showed, indeed, that developers toggle between the three comprehension models.

At any point during the comprehension process, any of the three components might be functional. During the creation of a program model, a developer might notice a beacon implying a common task, such as sorting. This leads to the theory that the code sorts something, allowing the top-down model to be invoked. The developer then creates subgoals and looks for hints to help those in the code. If the developer discovers some unnoticed code during this quest, he will return to building the program model. Structures created by one module can be accessed by the other two, but each module has its own set of desired knowledge types [28] [30].

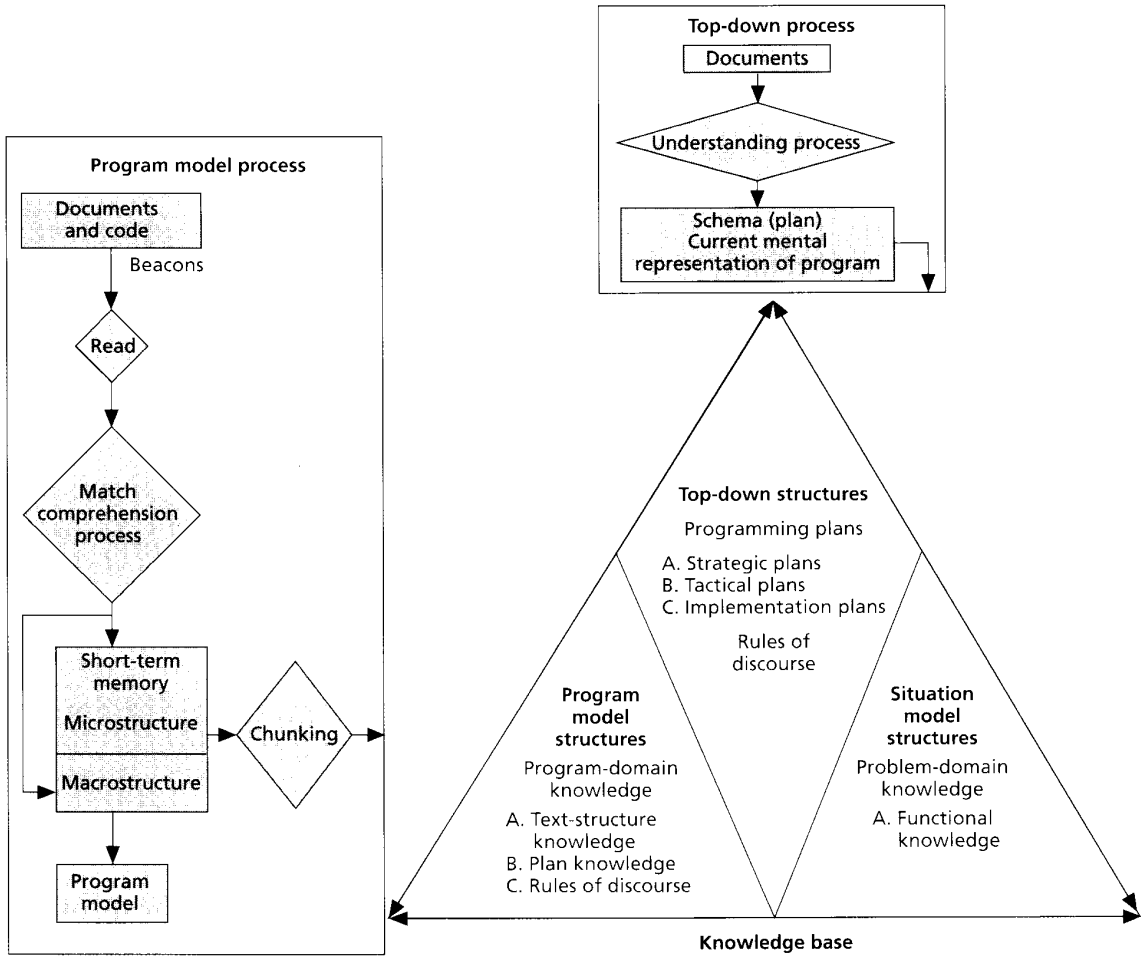


FIGURE 2.3: Integrated Metamodel [30]

.

Other notable comprehension models are:

- **Letovsky model:** The three key components of Letovsky's high-level comprehension model are a knowledge base, a conceptual model (internal representation), and an assimilation process. Programming skills, problem-domain understanding, discourse laws, strategies, and goals are all part of the knowledge base.

  an annotation layer, an implementation and a specification form a mental model. In the specification layer, the program objective is featured in detail by the program's highest abstraction level. Data structure and functions are represented

as entities in the implementation layer, which includes the lower-level abstraction [17].

- **Shneiderman and Mayer model:** The comprehension model of Shneiderman, and Mayer chunks the program in short-term memory into a semantic representation. These semantics has various program abstraction levels such as program goals and the algorithms that are used to achieve the program's objectives. Internal semantics development is aided by long-term memory, an information base of semantic and syntactic knowledge. Semantic knowledge is independent of any particular programming language and syntactic knowledge depends on knowledge in a specific programming language. program comprehension is a directional process it begins with source code and finishes once the problem is understood [31].

- **Brooks model:** In Brooks model, programmer domain knowledge is rebuilt with program comprehension. Understanding the problem is achieved by mapping knowledge in various domains. A top-down process is employed to build the mental model and the hypothesis are refined with a top-down approach iteratively considering various domain knowledge [32].

2.4 New approaches to measure Program Comprehension

According to researchers, new insights into the programmer can be obtained using neuro-imaging methods. When participants complete specified tasks, such as understanding source code, fMRI can be used to see which parts of the brain are triggered. Based on this model and more than 20 years of fMRI research, distinct brain regions are linked to different cognitive processes [33]. Some of the new approaches used to measure program comprehension in recent years are discussed below.

2.4.1 functional Magnetic Resonance Imaging (fMRI)

In the 1990s, researchers discovered that MRI(Magnetic Resonance Imaging) could be used to observe the changes in oxygen levels in the blood. There is a difference in the magnetic properties of oxygenated and deoxygenated blood, this can be captured with the help of BOLD signals using fMRI. Active regions in the brain require more oxygenated blood than the resting regions, these differences can be observed to precisely say which brain region is activated and cognitive processes associated with that brain region. In 1991 a study was conducted to map brain regions and cognitive processes [33].

Similarly, Siegmund, Janet, et al. used fMRI to study program comprehension. This is the first-ever study that used fMRI to study program comprehension and demonstrated the new approach and its potential in the study of program comprehension. As the subject is present in the fMRI machine, the code snippets are projected on a small mirror in the fMRI machine, the participants are required to provide the output of the snippet and identify syntax errors in the snippets. While the participants are comprehending the source code scan is performed to observe which brain regions are activated while code comprehension. Java programming language is used to develop the snippet for this study. A total of 17 participants are observed inside the fMRI scanner. The results of the study showed that During bottom-up program comprehension, Brodmann areas 6, 21, 40, 44, and 47 are activated and participants familiar with the Java programming language require less cognitive effort to understand the code snippets [4] [2] [34].

FIGURE 2.4: Experiment setup for use of fMRI in Program Comprehension [4]

.

In another study Siegmund, Janet, et al. conducted a study using fMRI to study the differences in bottom-up program comprehension and comprehension with semantic cues. Eleven participants to part in the study. As part of the study code snippets are presented to the participants which the participants are expected to solve. To induce bottom-up comprehension variable names are obfuscated in some code snippets, Semantic cues were used in some of the code, a well-printed layout was present in some code snippets, and in some snippets disrupted layout was used. The combination of all such constraints is presented to participants. The results of the study showed that comprehension of source code will be eased with the help of beacons [35].

2.4.2 Eye Tracking

Eye-tracking was first used in software engineering in the 1990s. Researchers study program comprehension, model comprehension, debugging, traceability, and

collaborative interaction using eye trackers. In program comprehension, eye-tracking is used to collect data on the cognitive processes of programmers. Eye trackers capture the eye movements of the participants, these eye movements provide valuable evidence of cognitive process in programmers' brain while they are solving certain tasks by seeing stimulus [36].

A variety of eye-tracking techniques exist to track the movement of the eyes. The three predominantly used techniques are video-based tracking, infrared pupil-corneal reflection, and Electrooculography-based tracking. video-based near-infrared light eye trackers have become popular nowadays. Infrared light is projected into the eyes of the subject and the light that hits the eye will be reflected and the reflected light is captured by the camera of the tracker. With the help of some calculation, the eye-tracker can precisely detect where the eye is focused [37].

Metrics used in Eye-tracking

Since 1990, numerous studies in software engineering have been conducted using eye-tracking in these studies researchers used various metrics. Some of the frequently used metrics in eye-tracking are presented here:

- **Fixation:** A fixation is an event in which the eyes are fixed on a certain part of the stimulus for over 200 to 300 ms. Fixation is the most commonly recorded event in Eye-Tracking data [38]. Fixation helps in deriving the information on the attention of a subject on the content of the stimuli. Fixation Count, Fixation Rate, Fixation Duration, and Fixation Time are the frequently used eye-tracking metrics related to Fixation.

- **Saccade:** Saccades are eye movements where the eyes move from one point of focus to another creating an attention path. Saccades are said to be very fast typically taking 30 — 80ms to complete. Saccades are not always seen to have a path between successive fixations but can move back and forth along the entire stimuli and these backward saccades are called regressions [39]. Information

processing only happens when the eyes are fixated but not during saccades motions, [40]. Number of Saccades, Saccade Duration, and Regressions rate are the frequently used Saccade metrics.

- **Pupil dilation**: In low light conditions, to allow more light into the eye the pupil gets widened this is called Pupil dilation. In program comprehension task pupil dilation occurs if the tasks is too complex to be understood [41].

- **Scanpath** : The participants eye movement pattern for the first fixation to the last fixation in chronological order is a Scanpath.

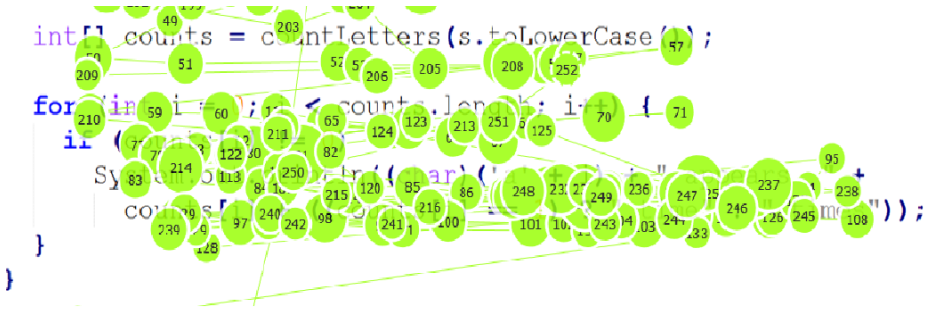


FIGURE 2.5: Fixations and Saccades [42]

.

Figure 2.5 represent the scan path on the program snippet. The circles in the figure represent fixations, the numbers on the circle represent the fixations order and the size of the circle represents fixation duration. Lines represent saccades [42].

2.4.3 Eye tracking studies in Software Engineering

Eye-tracking is been widely used in program comprehension in recent years. The first eye-tracking study in software engineering is carried out in 1990 by Crosby and Stelovsky, they studied the effect of the experience of programmers on comprehension. The study result found that programmer with less experience pay more attention to comments than experienced programmers [43] [44] [45].

Figure 2.6 shows the number of eye-tracking studies published in recent years. As it can be seen in the chart there is a sharp rise in publications in recent years.

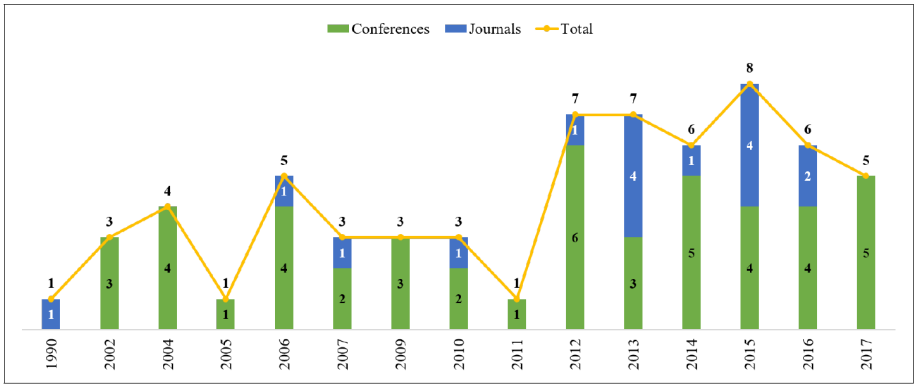


FIGURE 2.6: Number of Eye Tracking studies published in respective years [45]

.

Details of the few of the eye tracking studies in code comprehension

Code comprehension studies that used eye-tracking performed comprehension tasks, participants are expected to read and understand the programming task and answer a few questions regarding their understanding.

Following are a few of the code comprehension studies that used eye-tracking, details like the number of participants, programming language, type of eye tracker, variable are presented in the form of a table.

1. Crosby and Stelovsky studied the effect of experience on programmers code comprehension [43].

Artifacts	Pascal source codes of binary search algorithm.
Participants	18 students and faculty members
Eye-tracker	Not mentioned
DV	Number of fixations and time
IV	Expertise
MV	Not mentioned

FIGURE 2.7: Crosby and Stelovsky study on code comprehension [44]

.

2. Crosby et al. studied the difference in how beacons are used by the experts and novices to comprehend source code. The study found that the beacons simplify program comprehension [46].

Artifacts	Lines of code from binary search program written in Pascal and shown in random order.
Participants	18 students
Eye-tracker	ASL
DV	Accuracy and time
IV	Expertise and number of lines
MV	Not mentioned

FIGURE 2.8: Crosby.et.al study on code comprehension [44]

.

3. Bednarik and Tukiainen studied the programmers approach in using code and visualisation, the Java code was presented with help of a visualization technique [47]

Artifacts	Java source codes offactorial (15 LOC), recursivebinary-search (34 LOC),and naive string matching (38 LOC)
Participants	14 students
Eye-tracker	Tobii 1750
DV	Time and attention switching
IV	Code vs. visualization
MV	Not mentioned

FIGURE 2.9: Bednarik and Tukiainen study on code comprehension [44]

.

4. Busjahn et al. conducted an experiment in which the difference in the reading of source code and natural text was investigated [48].

Artifacts	Java source codes
Participants	15 participants
Eye-tracker	Tobii T120
DV	Time, number of characters, and number of elements
IV	Source code vs. natural language text, and different source code partsincluding: operator, keywords, identifiers, and numbers
MV	Not Mentioned

FIGURE 2.10: Busjahn et al study on code comprehension [44]

.

5. Sharafi et al. studied the effect of identifiers style: underscore vs. camel case on recall of identifier names by programmers [49].

Artifacts	Three Java sources code of 2D graphical frame (30 LOC), database tester (36 LOC), and nprime number calculator (44 LOC)
Participants	26 students
Eye-tracker	FaceLAB
DV	Accuracy, time, and effort
IV	Gender and identifier style (camel case vs underscore)
MV	Study level and style preferences

FIGURE 2.11: Sharafi et al. study on code comprehension [44]

.

6. Busjahn et al. studied the differences in attention distribution of experts and novices in reading strategies [50].

Artifacts	Eleven Java source codes
Participants	15 Professionals
Eye-tracker	Tobii T120
DV	Time
IV	Code elements (identifiers, operators, keywords, and literals)
MV	Expertise

FIGURE 2.12: Busjahn et al. study on code comprehension [44]

.

2.4.4 fMRI and Eye Tracking

fMRI has been used in neuroscience for several years and in software, engineering fMRI is been mainly used in program comprehension studies. Through the use of fMRI in program comprehension studies there are few limitations to this like in fMRI studies, for an individual task sequence of the mental process of a participant is not possible to observe. It is also difficult to say how the participants completed the task. There is a gap in understanding the individual program comprehension phases. The other limitation with fMRI is its temporal resolution is very low one to two seconds while the rapid cognitive process takes place in a fraction of a second.

To overcome these limitations of fMRI Peitek, Norman, et al. came up with the new approach of using fMRI and Eye-tracking simultaneously for measuring program comprehension. Such a combination of methods helps in overcoming the limitations and would complement each other's strengths. The information about the participants focus area and the high temporal resolution helps in identifying the brain activities precisely in time. With this approach, it would be easy to determine which cognitive process is triggered by what part of program stimuli. Peitek, Norman, et al were able to successfully conduct a program comprehension study by adding eye tracking to the fMRI experiment and proved that program comprehension measure with eye-tracking and fMRI is promising and also showed that fine-grained fMRI study is feasible with

simultaneous eye-tracking [51].

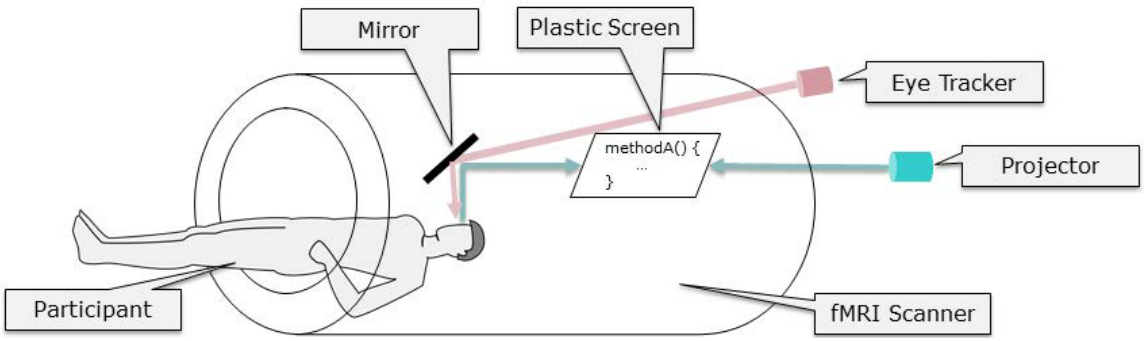


FIGURE 2.13: Simultaneous fMRI and Eye Tracking study setup [51]

.

Eye-tracking has been used in combination with neural measures like fMRI, EEG and fNIRS [52] [53] [54] but most of these studies used eye tracking as an indicator for whether the participant completed the task.

Other techniques used in program comprehension: Electroencephalography (EEG) is one of the approaches used to measure program comprehension. EEG measures the electrical impulses of a neuron's activation and, thus, has a high temporal resolution, but at the cost of lacking the high spatial resolution of fMRI. Moreover, extracting event-related brain activation with EEG requires much more averaging than in fMRI. Kluthe used EEG in program comprehension. He observed participants with different experience levels in the study and found that the participants with less experience felt the comprehension task was difficult to solve [55]. Functional near-infrared spectroscopy (fNIRS) is also been used in measuring program comprehension. It also measures the BOLD effect but it uses light-absorption properties of blood [56].

Chapter 3

Experiment Planning

Experiment planning is a critical aspect when performing experiments to ensure the experiment's success. Before experimenting, careful preparation is needed, which aids in the structuring of the experiment as well as the testing of the feasibility of the proposed research. Experiment planning could reveal design flaws, which could result in erroneous data.

The key aspects of planning and conceptualize an experiment are a well-defined goal of the research, well-designed experiment material to obtain desired results, participants with desirable qualities to suit the research, most impotently hypothesis and variable such as independent variables and dependent variables. The independent variables are the variables that are completely under the influence of the experimenter and can be manipulated according to the study's needs as well as the hypothesis. The dependent variables are the variables that we want to observe, these variables are the values obtained after experimenting hence can not be controlled and can be used to test the hypothesis [38]. Based on this hypothesis and variables, suitable tools and technologies are to be used to design an experiment. This chapter highlights other important aspects of experiment planning.

The experiment progress takes place in stages beginning with the development and selection of code snippets for the study as seen in Section 3.3 and ends with the evaluation and testing of the data that is collected to satisfy the formulated hypothesis. The stages of experimentation are described below:

1. **Stage 1:** The first stage is designing and choosing suitable code snippets to ensure that the experiment runs smoothly. The procedure for this operation is described in detail in Section 3.3. The next activity in this step is to get familiar with the SoSci Survey online platform since it is used to develop questionnaires and data collection in the form of an online survey. Another important activity is to learn how to conduct a study using the Think-aloud protocol. The details of the Survey set up using the SoSci survey platform and Think-aloud protocol are presented in later sections. Once the mentioned activities like code snippets creation, selection, and establishing understanding about the surveying platform and think-aloud protocol, experiment design move to the next stage.

2. **Stage 2:** In this stage questionnaire for the survey is to be created using the SoSci survey platform which involves a variety of steps ranging from generating individual questions to writing a questionnaire for data collection. The procedure for using the SoSci survey platform for creating questioners is described in section 3.6.1. Once the questionnaire is created it is important to test the questionnaire for the flow and errors. The collected sample date is also to be observed with respect to the questions in the questionnaires for any abnormalities.

3. **Stage 3:** Once the questionnaires are thoroughly tested for errors. The questionnaire is then ready for conducting the survey. Participants are invited to an online meeting and the link to the survey is provided to the participants after a short briefing about the survey. The time for the survey is chosen according to the participants' convenience.

4. **Stage 4:** Once the data is collected from the participants then data is cleaned for errors and deformities to ensure the formation of an accurate result set. Now the data set is ready for analysis to evaluate the research question and hypotheses. The data set can be visualized with graphs to give a clear pitcher of the obtained data.

The sub-sections contain details about the study's strategy and design. It contains

information about the hypothesis formed from the research question as well as how the experiment is expected to proceed to achieve the desired results.

Following are the details of the sub-section in respect to the above mentioned stages

1. **Sub-section 1** elaborates on the goal of the research.
2. **Sub-section 2** provides the details of the participants who took part in the experiment.
3. **Sub-section 3** provides details of the experiment material.
4. **Sub-section 4** describes the tasks that the participants have to deal with.
5. **Sub-section 5** describes the technologies that were used to build the experiment setup for data collection.
6. **Sub-section 6** provides the detail description of the experiment design along with details of the hypothesis formulated with respect to research questions, variables, and the survey setup in SoSci Survey.

3.1 Goals

The extensive research in the field of program comprehension since the past 30 years lead to the development of various comprehension theories, strategies, measuring techniques, and other valuable information about the comprehension of programs but still, there is a huge gap in understanding the cognitive process underneath program comprehension.

In present days with the advancement of technology researchers started exploring the feasibility of using fMRI, EEG, Eye-tracking, and a combination of these technologies to get better insights into programmers' brains.

Though the pandemic has a great influence on our lives restricting us to limiting our contact with peers and families. Pandemic also has a great influence on how research

studies have been carried out, once again with the help of technology it was possible to conduct the research remotely without risking the participant's health. The study is designed in such a way that the experimenter and participants may be in their desired location and yet produce valid data that can be used to form empirical evidence.

The main goal of this research is to determine whether there would be any differences in programmer's behavior while they are comprehending numbers and words algorithms. This behavioral difference could help in understanding the approaches and strategies of programmers while working with numbers and words. And also may lead to further research on behavioral differences.

This study considers numbers and words as a factor that affects the behavior of the programmer in terms of correctness and response time. This may help in understanding whether the number and words have an impact on program comprehension.

3.2 Participants

In this research, the effect of words and numbers on programmers behavior is studied with the help of data collected from external human subjects. The participation of the correct participants is an important factor for the study to generate useful and meaningful results. Non- suitable participants might result in unwanted results and even cause the failure of the study.

This study relies on the responses from the participants in the field of computer science. The link to the online survey is shared with the participants, before they start the survey the participants are briefed about what the task is and how to carry on with the survey.

Demographic data such as age, gender, education level, experience learning programming, experience learning the java programming language, and professional programming experience are collected. Though the participants involved in the study

have random programming experience, experience in java is important to cross-check the collected response with responses of participants from other programming domains.

Knowledge about the programming basics, data structures, algorithms, and the basic understanding of the java programming language is sufficient to answer the programming tasks in the survey. Knowing about the programming experience of the participants would help the researcher in better understanding of collected data. Preferred programming language and programming skills compared to participants' course mates and colleagues are the other demographic questions asked.

A computer, an uninterrupted internet connection, a web browser of participants choice like Chrome, safari Firefox, etc., and communication application such as Google meet, Microsoft Teams, Skype is the basic requirements from the participants end to take part in the survey.

To recruiter the participants, a description of the research topic and the background of code comprehension studies, and a request to participate in the survey are provided in the social media groups of my course mates and my bachelor's course mates. People interested and curious about the research took part in this research survey.

In Chapter 4 the further details about the participants who took part in this study and the distribution of data sample among the participants are presented.

3.3 Experiment Material

The experimental materials used in this study are code snippets. In a program comprehension study code snippets are an important part of the study, in most of the instances code snippets act as deciding factors for the experiment's success. Inappropriate code snippets for a specific purpose of the study would generate unwanted results [5]. This section briefs the creation and choosing of code snippets for this research. As the goal of the research is to find the behavioral differences of programmers in terms of correctness and response time while comprehending numbers and words

algorithms, this makes the selection of code snippets an important factor to evaluate the hypothesis.

3.3.1 Code Snippet Creation

Code snippets must be written following the study's design and documented in such a way that they can be easily modified and reused in other experiments and studies.

For this study, a total of 14 code snippets are created by taking inspiration from the previous studies on program comprehension such as Understanding Understanding Source Code with Functional Magnetic Resonance Imaging, Measuring Neural Efficiency of Program Comprehension, Simultaneous Measurement of Program Comprehension with fMRI, and Eye Tracking: A Case Study. Out of 14 code snippets, 10 snippets are finalized to be used in the study. Small to medium-sized code snippets with a maximum of 30 lines of code were used in the experiment.

The code snippets created for this study are designed in 5 sets, each set consists of two almost identical code snippets (i.e the programs are logically similar) one working on numbers and the other on words. For example, if we consider the algorithm removes duplicate one algorithm of the set removes the duplicate elements in a number array and the other removes the duplicates in the string array. Figures at the end of this section would explain this.

The code snippets are designed in such a way that all of the basic principles of the programming paradigm are considered, allowing the experiment to be more varied and challenging for the participants. The code snippets are created using Eclipse IDE. An Eclipse IDE plugin Check Style' is used to follow the coding conventions. All the created snippets are tested for errors and executed for cross-checking the outputs. The complexity of the snippets is taken into consideration so that the snippets do get too complex to be understood by the participants.

3.3.2 Code Snippets Selection Criteria

Once the code snippets are created the next step is to define the criteria to select the appropriate code snippets from the created set of snippets. The selection criteria would have a great impact on how well the participants carry out their tasks. A better understanding of potential participants for the study and the goal of the study is required to select appropriate code snippets for the study. Here are the criteria considered for the selection of the code snippets.

- The code snippets should be easy, The participants could also be novice programmers, if the snippets are complex to understand, the participants could take an unusual amount of time to answer or they may lose interest in the task leading to t experiment failure.
- The code snippets should be challenging enough for the participants so that relevant data can be generated, if the snippets are not logically challenging enough for the participants then they might solve the task without having a look at the snippet completely, this will generate an unwanted result.
- The snippets should represent the commonly known concepts that are taught during the initial stages of the programming course. More complex concepts might confuse the participants.
- Snippets should consist of various operation such as control structures and conditional statements.

Following are the two code snippets among the other eight snippets designed to be used in this research.

```java
import java.util.Arrays;

public class RemoveDuplicates {

    public static void main(String[] args) {
        int array[] = { 3, 7, 9, 9, 3 };
        int newArray[] = removeDuplicateNumber(array);
        for (int k = 0; k <= newArray.length - 1; k++) {
            if (newArray[k] == 0)
                break;
            System.out.println(newArray[k]);
        }
    }

    static int[] removeDuplicateNumber(int arr[]) {
        Arrays.sort(arr);
        int length = arr.length;
        int[] result = new int[length];
        int j = 0;
        for (int i = 0; i < length - 1; i++) {
            if (arr[i] != arr[i + 1]) {
                result[j] = arr[i];
                j = j + 1;
            }
        }
        result[j] = arr[length - 1];
        return result;
    }
}
```

FIGURE 3.1: Algorithm to remove duplicate elements in a number array

.

```java
import java.util.Arrays;

public class RemoveDuplicateWords {

    public static void main(String[] args) {
        String array[] = { "abc", "pqr", "xyz", "abc" };
        String newArray[] = removeDuplicateWords(array);
        for (int k = 0; k <= newArray.length - 1; k++) {
            if (newArray[k] == null)
                break;
            System.out.println(newArray[k]);
        }
    }

    public static String[] removeDuplicateWords(String words[]) {
        Arrays.sort(words);
        int length = words.length;
        String[] result = new String[length];
        int j = 0;
        for (int i = 0; i < length - 1; i++) {
            if (words[i] != words[i + 1]) {
                result[j] = words[i];
                j = j + 1;
            }
        }
        result[j] = words[length - 1];
        return result;
    }
}
```

FIGURE 3.2: Algorithm to remove duplicate elements in a string array

.

As it can be seen in Figures 3.1 and 3.2 both the algorithms are to remove the duplicate elements from an array and they are almost identical in their programming logic. One algorithm works with an integer array and the other works with a string array.

3.4 Tasks

In most of the program comprehension studies, comprehension tasks and debugging tasks are the common sorts of tasks that the participants are expected to solve.

In comprehension task participants are provided with code snippets and asked to determine the output of the program. The snippets used are usually that perform simple integer arithmetic, string operations like reversing a string, concatenating strings, etc, searching and sorting algorithms like binary search, bubble sort, etc.

Sometimes constraints like obfuscation and beacons are used to induce bottom-up and top-down comprehension respectively [35] [4]..

In debugging or syntax tasks, syntax errors like missing parenthesis, miss-matched quotation marks, missing identifiers, and semicolons are introduced into the code snippets participants are asked to identify the errors in the given code snippets.

In this study, like the above-mentioned studies comprehension tasks are used to study the behavior of the participants. Participants are provided with 4 code snippets that perform arithmetic operations on numbers, search operation on a string for a specific character, integer, and string array manipulations. Participants are asked to go through the code snippets that are provided to them and indicate the out of each algorithm. Apart from the comprehension task the other important task for the participants is all the participants are asked to say whatever they are thinking out aloud. As the participants are going through the snippets and understanding the logic they are asked to verbalize their thoughts so that a clear understanding of their approach to solve the tasks can be gained. Further details about how the snippets are presented to the participants, how the participants provided their inputs, and how the data is collected are discussed in the Experiment Design section.

3.5 Tools and Technologies used

The implementation of the experiment plane involves the use of various tools and technologies combination. Each tool or technology has a distinct role to fulfill, and they operate at various stages of experiment implementation. The following are some of the most relevant tools and technologies used in this study:

1. SoSci Survey Tool: SoSci Survey is a specialized online questionnaire tool for conducting scientific online surveys. This is an online tool that runs on the SoSci server and can be accessed via a browser. With premium tool services, this tool may also be set up on private servers of businesses or universities based on their

needs. When research or thesis is not performed to generate business or any other monetary gain and the research data is available to the public, then a student or a researcher of the university or any non-profit organization can use the services of the SoSci for free. SoSci survey can be used by a variety of professionals for a variety of reasons, including marketing professionals from businesses and university students and researchers for research purposes. Following are the features that are available to university students for free [57].

- The tool includes custom questionnaire filters that enable students to impose custom constraints on the survey process and the types of responses that are expected.
- Languages such as HTML, JavaScript, and PHP are integrated into the platform. These programming languages would help the researcher to customize the pattern of the question and the order in which the questionnaire is to appear.
- If a random set of questions (e.g random set of code snippets) are to be provided to the participants, then by applying PHP code randomization of questionnaires can be achieved.
- With UTF-8 encoding, the questionnaire supports multiple languages and special characters from various languages like German, Russian, Korean, etc.
- SoSci Survey provides a section where all the collected data can be viewed and downloaded. The collected data can also be filtered with various constraints to view specific details of the collected data.

With such a wide variety of questionnaire setup options, features, and flexibility that a tool has to offer, the SoSci survey is one of the popularly used online survey tools used by researchers. The setup of the survey questionnaires for this research

using the SoSci Survey are detailed described and illustrated in the 'Experiment Design' section.

2. Other tools used for this study are OBS, Skype, Google meet, and Microsoft teams. As it can be seen the last three tools that are mentioned here are used for online communication, the survey is conducted online with the help of these tools. On the other hand, OBS (Open Broadcaster Software) is an open-source streaming and recording tool. It provides features like screen capturing and audio recording. OBS is used to record the screens and voices of the participants while they are participating in the survey. Few of the communication tools mentioned above such as google meet and Microsoft teams do not allow screen recordings in their free versions to compensate for this OBS is used in this study. The detailed setup of the OBS software to capture screen and audio is presented in the 'Experiment design' section.

3.6 Experiment Design

With the focus on the aim of this study to determine how numbers and words affect programmers behavior in terms of time and correctness during program comprehensions, the variables of the study are described here:

- Dependent Variables
    1. Result data: The resultant data consist of participants behavior like time take to respond to the question and correctness of the answer while they are working with words and numbers algorithms.
- Independent Variables
    1. Code Snippets: The algorithms working with numbers and words are the same in their programming logic.

2. Programming Experience: Participants in this study are programmers with varying levels of programming experience.

3. Programming Language: All the code snippets used in the study are developed in java programming language.

The purpose of this study can be formulated into a research question such as,

- **RQ1:** Does the correctness of programmers differ for numbers and words algorithms?
- **RQ2:** Does the response time of programmers differ numbers and words algorithms?

The above mentioned research question can be formulated into two pairs of hypotheses to test the collected data sets, the hypotheses are listed below:

- $H_0(1)$: Correctness of programmers does not differ for algorithms working with numbers and algorithms working with words.
- $H_1(1)$: Correctness of programmers differ for algorithms working with numbers and algorithms working with words.
- $H_0(2)$: Response time of programmers does not differ for algorithms working with numbers and algorithms working with words.
- $H_1(2)$: Response time of programmers differ for algorithms working with numbers and algorithms working with words.

3.6.1 Experiment setup

This section provides the details of the tool setup to conduct the experiment, first the detailed questionnaire design in the SoSci survey platform is described and then the detailed setup of OBS recording tools is presented.

- **Setting up SoSci Survey:** SoSci Survey is a simple tool for designing and building questionnaires with customizable questions. Selecting suitable question and answer types and writing questionnaires using drag-and-drop operations can be used to generate surveys.

  In this thesis the difference in the correctness and response time of the programmer while they are comprehending number and words algorithms is studied, The questionnaire is designed to gather information about the participants understanding of code snippets, so the participants are asked to provide the output of the code snippets that they went through and level of confidence in their answer. Other questions regarding participants' education level, current profession, experience in programming, professional experience, and preferred programming language are asked to learn more about the participants who took part in the survey.

  The layout of the SoSci survey platform can be seen in the figure 3.3. Layout in the figure 3.3 appears once a user login into their account. The steps to create a questionnaire are as follows

  1. Complete the login process and pick the desired project from the "Projects" tab's list of projects.

  2. If a project is selected, a page is loaded that displays the controls for creating survey questionnaires as seen in the figure 3.3. There are three key sections: The first control section to the left top corner is 'List of Questions' this section provides the option to create sets of questions, questions can be grouped into sections which helps in properly organizing the questions to creating the questions. All the questions that are created for the study can be seen here in this section.

     The next section is 'Questionnaire' here all the features such as 'Compose Questionnaire', 'images and media files', Questionnaire layout', 'Special

Features', etc that are related to creating a questionnaire can be found. Using these features a questionnaire can be created, image files and media files required for the questionnaires can be uploaded and the theme of the questionnaire layout can be changed, also footer text can be edited and a logo can be added.

The next section is 'Controls', here all the settings related to a project can be found. In this section one can exercises features like an overview of the project, project setting for how long the project be active, test the project, preview, and even download the collected data sets.

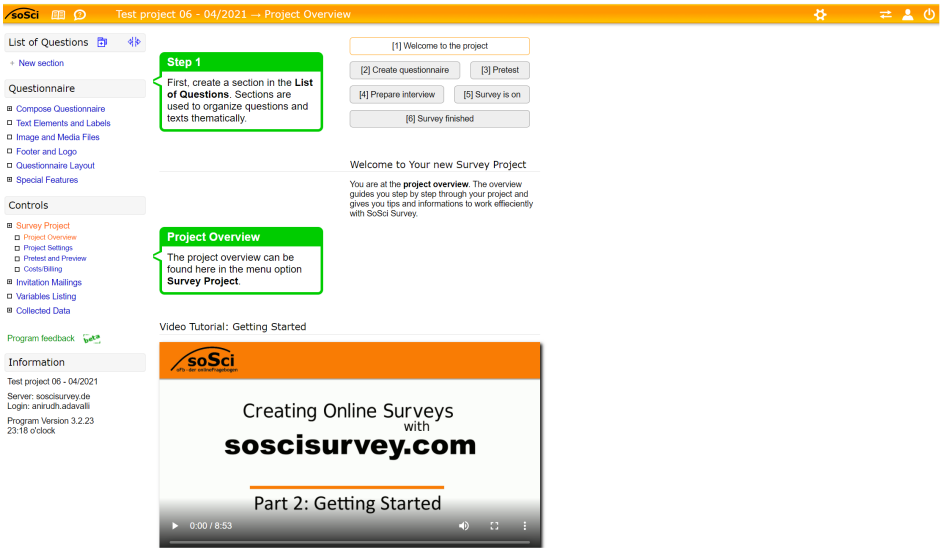


FIGURE 3.3: SoSci Survey online platform showing various controls

.

3. In this study, two different question sections are created in the SoSci survey platform one for the questions related to the comprehension tasks and the other for the demographic questions. This can be observed in the figure 3.4. The comprehension task section consists of an image that shows code snippets the participants have to understand, a text area to provide an answer to the snippet, and a question to indicate participants confidence in the given answer. These questions can be created in the 'list of questions' section by clicking 'add question' and 'add text', in this subsection various

types of question layout can be selected, questions can be composed and images can be shown in the questions with the help of 'add text' option. Images can be simply added to the question by adding HTML of the image to be presented to participants is provided in the 'add text' option. One such text creation with the image inserted can be seen in figure 3.5

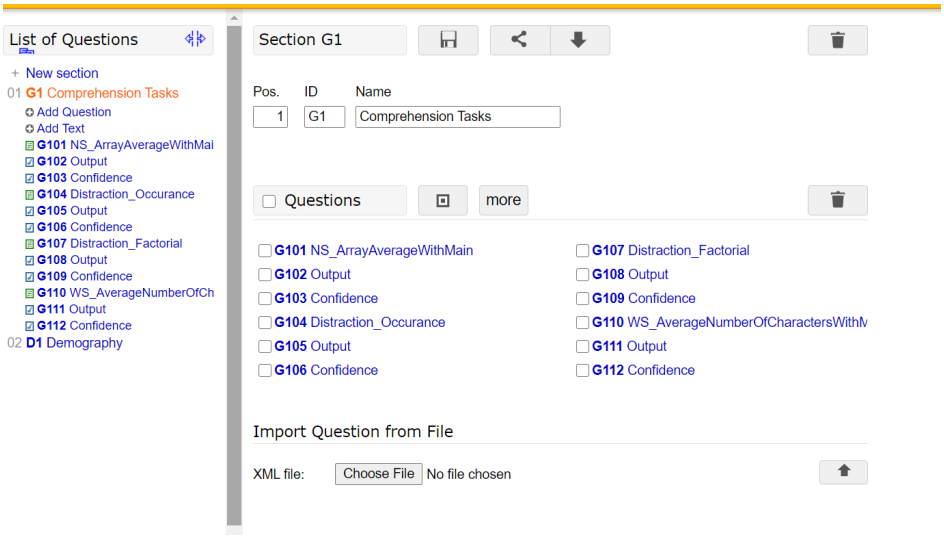


FIGURE 3.4: The list to questions section showing two different section of questions 'Comprehension Task' and 'Demography'

.

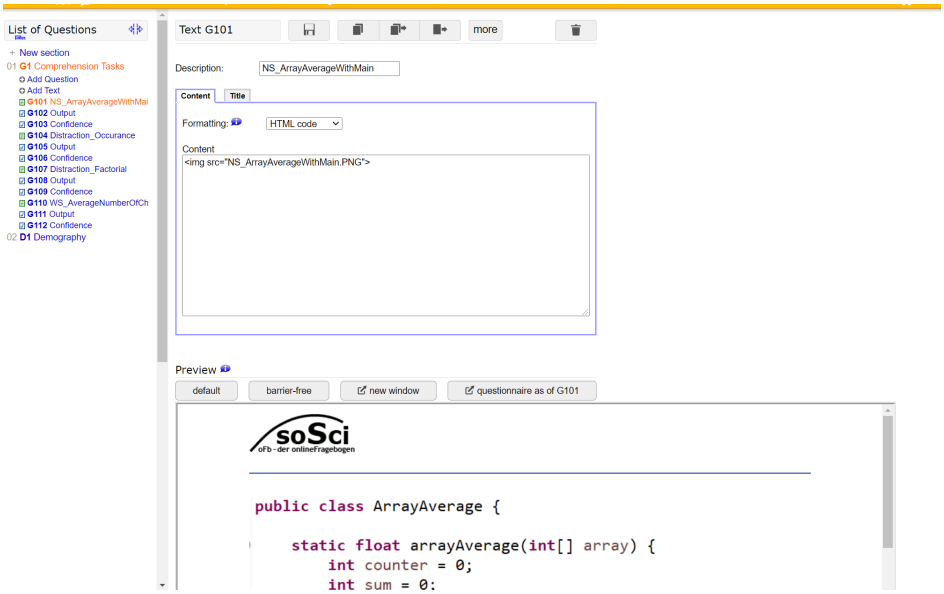


FIGURE 3.5: Figure showing a created question

.

4. Once the questions are created, the questionnaire can be created with the help of the 'Questionnaire' section. With the 'Create a new questionnaire' option a questionnaire template can be created which can be seen in the figure 3.6. Questionnaire Pages' is the tab the user is presented with. In this tab there are provisions to create pages for the questionnaire, each page has a page number and the id of the page, the questions created earlier will appear in the right bottom corner of the user interface. Desired questions can be dragged and dropped onto the page. In the study the questionnaire is designed in such a way that the first page of the questionnaire asked for the participants consent regarding data collection thereafter each page presents the participants with a code snippet, a provision to write the output of the snippet, and a confidence question, there are 4 pages with this configuration and then the 6th page presents the participants with the demographic question and in the last page, the participants are thanked for their participation and spending their valuable time. An example questionnaire is provided in the Appendix section of this report.

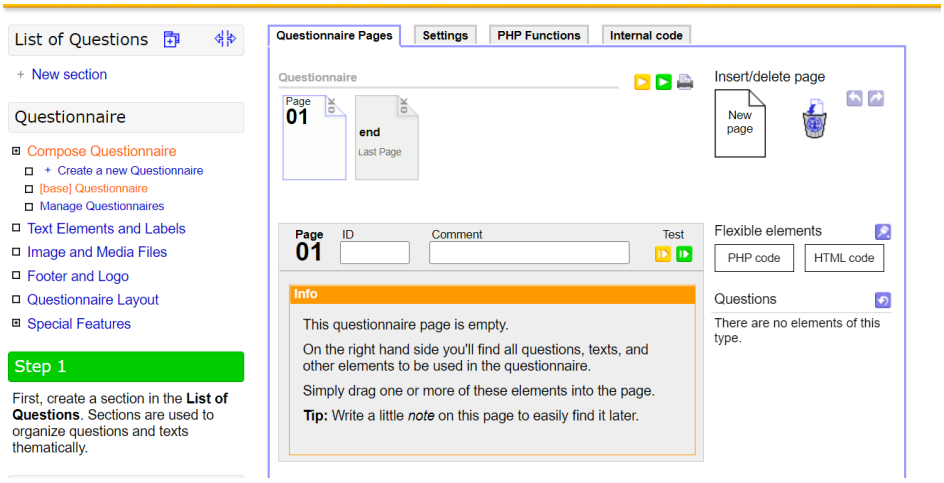


FIGURE 3.6: Feature to create a questionnaire

.

5. Once the questionnaire is prepared, it can be previewed, the yellow and green play buttons enable the preview of the questionnaire. The yellow button enables the preview in debug mode. In debugging mode where the

question id of each question on a page can be seen along with some additional information. The green button can be used to preview the questionnaires as it appears to the participants. Preview helps in view the layout of the questionnaire and detect mistakes in the questionnaire beforehand.

6. As the questionnaire is prepared and previewed the 'controls' section of the SoSci survey provides some more options to manage the project. 'Project Overview' is the option that enables the users to see the status of the project whether the survey is on or finished. The 'Project settings' section provides an option to edit the basic information about the project and set the survey administration period and enable to export of the project. The 'Pretest and preview' option enables the pretest of the survey and preview.

7. Another important section is 'Collected data' this section provided all the data that is collected from the participants. There are various option to collect different sets of collected data

   Once all the questions and questionnaires are created and tested the survey is ready to be presented to participants. In the 'manage questionnaire' section links to all the questionnaires are available and these links can be provided to the participants to take part in the survey.

- **OBS:** Open Broadcaster Software is a screen and audio recording tool. Some of the communication application like Google meets and Microsoft Teams does not allow the recording of the meetings while using them for personal use. Half of the participants preferred to use these applications for the meetings, so to record their screen and audio OBS is used. Following are the steps to set up OBS:

  1. Download the Software for the official website https://obsproject.com/ for the respective operating systems.

  2. Click on the downloaded file to install the software and follow the onscreen instructions.

3. The OBS application interface is shown in the figure 3.7 The application shown in the figure 3.7 is already configured and ready to be used. As it can be seen in the figure there several mini tabs at the bottom of the screen like 'Scene', 'Sources', 'Controls' etc. Click on the (+) symbol in the 'sources' tab to see more options and select 'Display capture' and 'Audio Output Capture'. Once these options are chosen the tool is ready to be used. By clicking the center portion on the screen, the size of the screen to be captured can be adjusted [58].

4. The 'Controls' mini tab provides various buttons like 'Start Streaming', 'Start Recording', etc. To record the meeting click on the 'start recording' button and once the meeting is ended click on the 'stop recording' button. The recorded meetings can be found by navigating to 'Files' -> 'Show Recordings' [58]

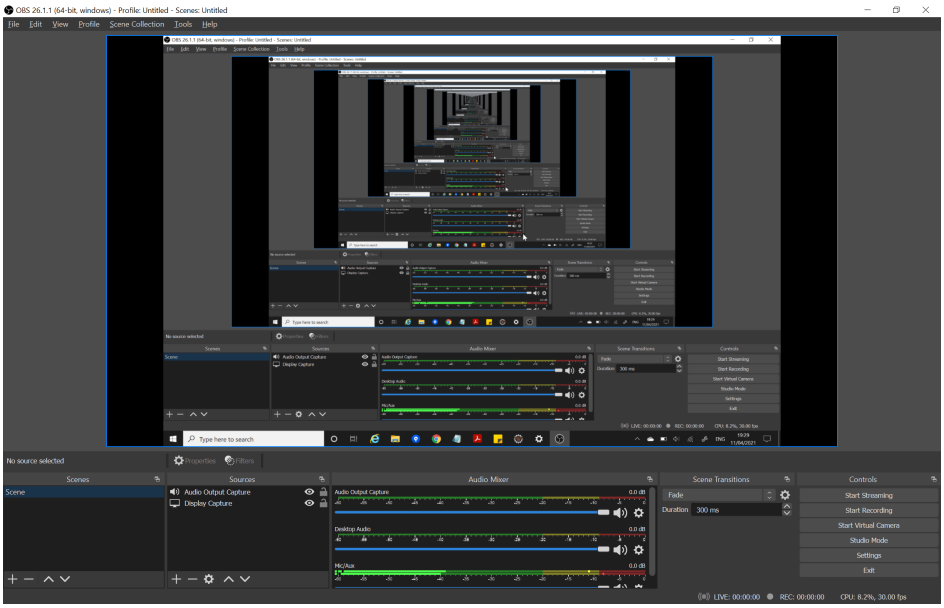


FIGURE 3.7: OBS tool user interface

.

3.7 Analysis Procedure

The collected data is subjected to statistical analysis to determine its significance. Statistical analysis algorithms are used to determine the importance of collected data sets. The T-Test, Mann-Whitney test, Needleman-Wunsch Algorithm, and other algorithms can be used to test significance. In this study, the Mann-Whitney test is used to test the significance of the data obtained from the survey. Mann-Whitney test is used to compare two independent samples for differences in them. This test is quite suitable for small sample sizes and the sample size can different for two samples for comparison. This test is also widely used in testing the significance of the hypothesis, these are the reasons for choosing the Mann-Whitney test for testing the hypothesis for this study [59].

Following are the steps to conduct Mann-whitney test:

- The first step is to determine the hypothesis for the given research questions. Each research question will have two hypotheses, the null hypothesis which is denoted by $\mathbf{H_0}$ and the alternative hypothesis is denoted by $\mathbf{H_1}$. For instance, if the difference between the two samples is to be determined then the Null hypothesis ($\mathbf{H_0}$) states that there is no difference between the two samples and the alternative hypothesis states that there are differences between the samples [60].

- Once the hypotheses are determined, the data samples for two different treatments are sorted from lower to higher values and are ranked from 1 to n where n is the addition of two sample sizes. If there is any tie in the sample data then the average of the ranks for the tied samples are calculated and assigned to tied samples [60].

- Once the rankings of the sample data are done, these rankings are to be added for two sample groups. The Sum of the rankings is used to calculate U_{Stat}. Formula

to calculate U_{Stat} are provided below. The lowest value obtained among the two sample groups will be chosen as U_{Stat} [60].

- Once the U_{Stat} is calculated, it has to be compared with the $U_{Critical}$ which can be found in the Mann-Whitney test critical value table for various significance values denoted by α. If the U_{Stat} is greater than $U_{Critical}$ then the null hypothesis should be accepted else it should be rejected.

Following is the formula to calculate U value:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2}$$

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2}$$

Here U_1 is for the sample one and U_2 is for sample two, the smallest value among these both will be U_{Stat}. R_1 and R_2 are the sum of the ranks of sample one and sample two respectively. n_1 and n_2 are total number of ranks of sample one and sample two respectively.

Chapter 4

Conduct

Chapter 3 provided the detailed implementation plan and design of the experiment. In this chapter, a detailed description of the experiment implementation and data collection is presented. This chapter is divided into three sections:

- The first section describes the participants briefing.
- The second section presents the procedure followed to implement the experiment.
- The third section provides the details of data collection like how and where the data is collected, what tools are used to collect data.

4.1 Participants Briefing

After the questionnaire has been thoroughly tested, it is ready for the actual survey. The survey will start with a short briefing. The participants must be briefed with instructions on what is the survey about, The procedure involved in the survey, What are the tasks they are expected to solve, things that they should keep in mind while taking the survey and some general instructions to take the survey correctly. The briefing is very important because it helps in carrying out the study as planned so that no error data is produced and the experiment does not fail.

The list of instruction that the participants are expected to follow are :

- The participants are requested to dedicate 30 minutes of their time without any interruptions during the survey.

- The participants are asked to speak their minds while they are solving the task. They are requested to be loud and clear to record their voices.
- The Participants are asked to share their screens while they are dealing with the tasks. They are informed about their screen will be recorded along with their voice.
- The participants are asked to focus completely on the task at hand, understand the snippets thoroughly and then answer.
- Participants must consider the importance of the study and try to answer all the questions properly such that valuable data can be collected.

4.2 Procedure

In this section, a detailed description of what happens to the participants from the moment they start the questionnaire to the point they finish it.

1. The first task is to invite the participants to take part in the survey. The Participants interested in the research responded with their willingness to take the survey. Each participant is suggested to choose a time slot of the day that they prefer. Participants provide their availability to take the survey according to their convenience.
2. On the scheduled date and time an online meeting is organized. The online meeting is conducted with the help of communication tools such as Google meet, Microsoft Teams, and Skype according to users convenience. In a normal circumstance the experiment would have taken place face to face but due to the Covid-19 pandemic, keeping the participants well being and restrictions in mind all the meeting sessions had to be conducted online.
3. As the meeting starts, the participant is given a briefing as mentioned in the 4.1. Details about the survey, the kind of tasks he/she is expected to solve, a short

description of the think-aloud protocol, and the task of verbalizing their thoughts while dealing with the tasks are clearly explained to the participant.

4. Once the participant is ready for the survey after the briefing, the participant is asked to share the screen so that his/her activities during the survey can be seen. The meeting session will be recorded, if the recording option is not available then the screen capturing software OBS is used to record the session.

5. Once the participant start sharing the screen a link to the online survey is shared with the participant through the chat window of the meeting tool. The questionnaire opens in the new window as soon as the participant clicks the link.

6. On the first page of the questionnaire, the participant encounters a consent question whether he/she agrees to take part in the survey along with the short description of the objective of the study and what task lies ahead in the questionnaire.

7. As the participant gives his/her consent, from the second page till the fifth page the participant would face comprehension tasks. Each page consists of an image of the code snippets designed and selected in the section 3.3 along with a provision to provide the output of the code snippet and his/her confidence in the provided answer.

8. Comprehension tasks are presented in this order from page two till page five, the first comprehension task would be an algorithm that works with numbers (numbers algorithm) or the algorithm that works with words (words algorithms), then two distraction snippets are presented in two consecutive pages to distract the participants and then again a numbers algorithm or a words algorithm. As mentioned in the variables of the study both the words and numbers algorithms should be identical in their logic. For example, if the first task is removing duplicated from an integer array then the last task would be removing duplicated

from a string array and if the first task is a words algorithm then the last would be a numbers algorithm with the same programming logic.

9. From page two, while the participant tries to comprehend the tasks he/she would verbalize his/her thought process and provide the output of the code snippet. This will continue till page five and then a set of demographic questions are presented to be answered on page six and then the questionnaire ends on page seven with a thank you note.

10. As the last step of the study, the participants are asked few questions about their approach in solving the comprehension tasks as part of the think-aloud protocol. Following are the questions asked at the end of the survey:

    - When you first saw the algorithm, what drew your attention?
    - Did the names of the variables, functions and class name helped you to figure out what the algorithm is about?
    - Did you find any similarities in the algorithms?
    - Did it make a difference to you if the algorithm used integers or strings?
    - Was your approach to solve numbers and words algorithms different?

    With this, the survey will end. The recording will be stopped and the recorded meeting session will be stored on the local system.

The sample of the questionnaire will be provided in the Appendix section of this thesis report.

4.3 Data Collection

The survey was conducted over for a month, participants are available to take part in the survey on different days and times. The survey's response data is automatically entered into the SoSci survey tool and can be found in the "Collected Data" section.

The SoSci survey platform provided various options related to collected data. The "View Data Set" option provides all the data that is collected and other several options such as "filter" can be used to filter the collected data by questionnaire. The data is presented in the form of a table, each response data set collected can be identified with a "Case Number" which is automatically generated by the SoSci Survey platform. Each data set can be expanded in a new window by clicking the case number, an exact questionnaire that the participants answered including the start time can be viewed. The actual response provided by the participants can be viewed.

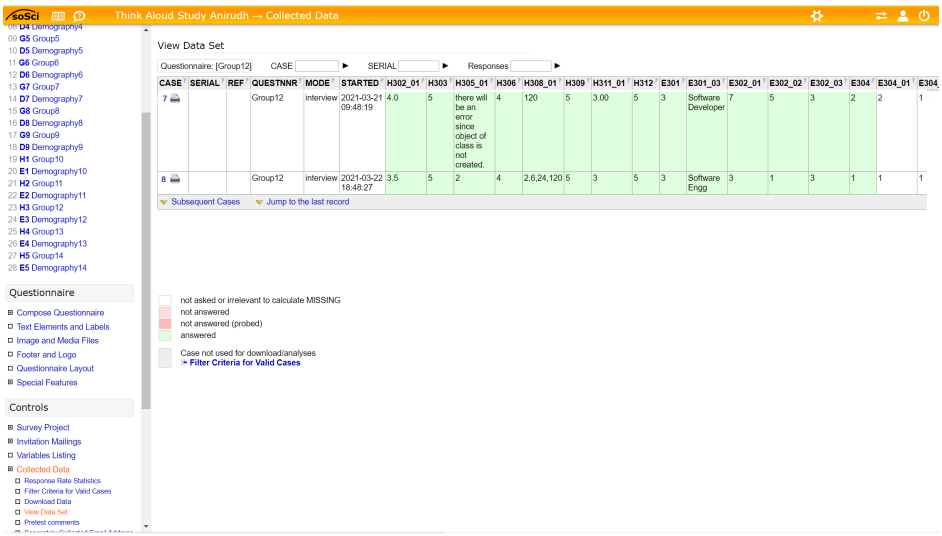


FIGURE 4.1: "View Data Set" option presenting the collected data

.

As it can be seen in the figure 4.1 a table with responses for a specific questionnaire is presented with the "View Data Set" option. Vivid ranges of options are available to view the data like responses from a specific questionnaire can be selected, responses can be filter based on the case, serial, and responses. The responses table can be navigated with options like "Jump to the first record", "Previous Cases", "Subsequent Cases", and "Jump to the last record".

A total of sixteen questions in a questionnaire are presented to the participants. Participants are required to answer all these questions to complete the questionnaire.

The questions in the questionnaire can be divided into the following categorized:

1. The first question is to obtain participants consent to take part in the study.
2. **Comprehension tasks:** There are eight questions that are related to comprehension tasks, participants are required to verbalize their thoughts while understanding each task and provided the output of the code snippet and their confidence in the provided answers.
3. **Demographic Questions:** In this four-question series, participants are asked about their gender, age, education level, and current occupation.
4. **Experience, skill, and preference:** Three questions related to participates experience in learning programming, programming in java, programming professionally, and preferred programming language are asked.

A total of 15 participates took part in the study. Responses from only 12 participants are considered in this study, The demographic data of the participants can be seen in the figure 4.2. The collected responses are distributed among the participants of different genders, age groups, and education levels. Eight participants are male, four participants are female. Five participants have a master's degree and seven participants have a bachelor's degree in the field of Computer Science. Eight of them are currently working as software Engineers and four of them are pursuing a master's degree in Computer Science. The answers provided by the participants to the code snippets and the time taken by the participates to complete each comprehension task are recorded and can be accessed from the "Collected Data" option in the SoSci Survey platform. As part of the think-aloud protocol, all the meeting sections are recorded. The audio recordings are converted into transcripts to understand the difference in participants approach for numbers and words algorithms.

The data that is collected from the code shippers is more important to this study rather than the demographic data of the participants. So demographic data of the participants can be considered as an informative part of the study. To gain a better

perspective of the participants questions related to their experience and preferences are asked.

Distribution	Participants Data
Male	8/12 (66.6%)
Female	4/12 (33.4%)
Average Age (Years)	28
Average Experience in Java (Years)	1.3
Average Professional Experince (Years)	2.6

FIGURE 4.2: Participants Demographic Data

.

The data collected as part of the think-aloud protocol such as the participants audio while they are verbalizing their thoughts and their answers to the questions at the end of the survey are converted into transcripts. These transcripts are also the major source of information regarding participant's approach and thought process while they are understanding numbers and words algorithms.

Chapter 5

Data Analysis and Results

In the last chapter implementation of the experiment and the data collection is discussed. In this section data cleaning, and analysis of data will be presented.

5.1 Data Set Preparation

Data set preparation is an important task. Any incorrect and erroneous data should be removed to gain accurate results. The data set is also to be formulated into analyzable data sets. These formulated data sets will help in evaluating the hypothesis more easily. The raw data collected from the study consist of inadequate responses from a few of the participants. This led to the removal of the three responses for data analysis. As mentioned in the section 4.3 a total of 12 responses from the participants are considered for analysis and evaluation, the rest of the three participants responded vaguely and without full attention to the logic of the code snippets.

The data obtained from the survey is organized into various types of data sets, which are then analyzed to produce acceptable scientific conclusions. The formulated data sets are organized into multiple table data sets that display various types of data values, and they can be categorized as follows:

- **Correctness of Numbers and Words Algorithms Data Table:** This data table consist of number of correct answers provided by the participants to each numbers and words algorithm.

- **Response time of Numbers Algorithms Data Table:** This data table consist of average time take by participants to answer the number algorithm snippet.
- **Response time of Words Algorithms Data Table:** This data table consist of average time take by participants to answer the words algorithm snippet.

The above-mentioned tables will be presented in section 5.3 of this chapter. Along with the responses to comprehension tasks, results from the transcripts of participant's audio which are recorded as part of the think-aloud protocol are also provided. The audio recordings of the meeting sections are used to generate transcripts. A tool called 'otter' is used to convert the audio files into text transcripts. This tool provided various options to edit the text that is generated during the audio-to-text conversion. The generated transcripts are cleaned to remove noise with the help of the otter.ai tool. [61]

5.2 Hypothesis Testing

For this research, statistical analysis is essential to see whether the formulated data can be tested to determine the significance value. Although the sample size is limited, this statistical calculation can be used as a model for future studies in a similar research domain [62].

The data tables in the section 5.3.1 i.e "Correctness of Numbers and Words Algorithms Data Tables", "Response time of Numbers Algorithms Data Table" and "Response time of words Algorithms Data Table" which represents the correctness of answers provided by the participants and response time are used for statistical evaluation. The hypotheses stated in the section 3.6 are tested for significance using the Mann-Whitney U test. The results of the test are presented here:

- **Correctness of answers:** The data collected from 12 participants is used to conduct the test. The difference in the correctness of answers presented by the

participants are tested against numbers and words algorithms is tested. This test is to answer the research question one **RQ1**. The results of the test are as following: U-value = 72, $U_{Critical}$ = 37 at a significance value $\alpha = 0.05$, Z-Score = 0.02887 and p-Value = 0.97606. The U-value is greater than $U_{Critical}$ which implies the null hypothesis $H_0(1)$ is accepted. The results state the there is no difference in the correctness of answers for numbers and words algorithms.

- **Response Time**: The response time of twelve participants is used to test the hypothesis. The difference in the response time of the participants for numbers and words algorithms is tested. The hypothesis formulated for the research question **RQ2** is tested. The test resulted in the following results: U-value = 62, $U_{Critical}$ = 37 at a significance value $\alpha = 0.05$, Z-Score = -0.54848 and p-Value = 0.58232. The U-value is greater than $U_{Critical}$ that implies the null hypothesis $H_0(2)$ is accepted i.e there is no difference in the response time for numbers and words algorithms.

5.3 Results

In this section, the result set is explained in detail and also the insights obtained from data analysis are provided. Appropriate conclusions can be drawn from the results obtained by the implementation of the study. Based on formulated data sets it can be assumed that the resulting data is satisfactory. Any potential inaccuracies in the results will be discussed, but that is a topic for the next chapter, which will address the possible inaccuracies and logical errors to justify.

This section is divided in to two subsections as listed below:

- The first subsection explains the data in the formulated data tables.
- The second subsection will provide the visual representation of the results.

5.3.1 Data

The data generated as a result of the experiment is interpreted in a way that shows valuable insights which help answer the research question. As mentioned in section 5.1 the data collected for the survey is formulated into a table. These data would help in justifying the research questions. The data from the different data tables are interpreted as follows:

- **Correctness of Numbers and Words Algorithms Data Table:** This data table presents the total number of correct answers provided by the participants for questions related to numbers and words algorithms. It can be seen in the figure 5.1, out of twenty four responses a total of twenty two correct responses are provided by the participants and the total number of correct answers accounts for 91.6 percent.

Code Snippet	Total correct answers for each algorithm (out of 4)
ArrayAverage	3
ClosestValue	4
RemoveDuplicates	4
AverageNumberOfCharacters	3
WordsClosestValue	4
RemoveDuplicateWords	4
Summary	22 (91.6%)

FIGURE 5.1: Data table showing the number of correct answers provided by participants for each algorithm

.

- **Response time of Numbers Algorithms Data Table:** This table consist of information regarding the average time take by the participants to respond to the each numbers algorithm. The figure 5.2 shows that an average of 291 seconds are taken by the participants to answer each numbers algorithm.

Code Snippet	Response time in seconds
ArrayAverage	202
ClosestValue	335
RemoveDuplicates	335
Average	290

FIGURE 5.2: Data table showing the average time taken by the participants to answer each numbers algorithm

.

- **Response time of Words Algorithms Data Table:** This table consists of information regarding the average time taken by the participants to solve words algorithms. It can be seen in the figure 5.3 an average of 294 seconds are taken by the participants to answer each words algorithm.

Code Snippet	Response time in seconds
ArrayAverage	227
ClosestValue	315
RemoveDuplicates	341
Average	294

FIGURE 5.3: Data table showing the average time taken by the participants to answer each words algorithm

.

Apart from the responses to the questionnaire i.e response to the comprehension tasks and demographic questions the other important data source from the survey is the transcripts of the participants audio. As mentioned in the section 4.2 after each survey a short interview is conducted, in which participants are asked few questions about various aspects of the task that they have solved. Below is the summary of the results that are gathered from the transcripts developed using audio files.

The following insights are obtained from reading the transcripts produced from the audio of the participants:

- As soon as the participants saw the code snippets, they began looking at the names of the identifiers, functions, and classes to find out how the code snippet worked. All of the participants were looking for function calls, variable declarations, the function's return type, and the function body at first glance and were vaguely going over the entire snippet, and once they had that detail, they began going over the function body, loops, and statements in the function body. All the participants have taken this approach.
- Participants with prior java programming experience were able to complete the loops and statements much more quickly and easily than those with little or no prior experience.
- Participants with less experience went through the snippets line by line, while those with more experience were able to skip certain loops and correctly answer the questions. Also, experienced participants took less time to comprehend the snippet than those with no or limited experience.
- The way the participants attempted to solve the number and word algorithms (code snippets in the survey) were identical. To solve the number and word algorithms, they used the same method as mentioned above.

The participants answers to the questions listed in section 4.2 are given here as part of their response to the short interview questions after the survey:

1. **When you first saw an algorithm, what drew your attention?** All the participants responded that the class name, function name, function return type, identifier name caught their eye at the first glance.
2. **Did the names of the variables, functions, and class name helped you to figure out what the algorithm is about?** All the participants mentioned that the names of the class, functions, and identifiers are meaningfully and

self-explanatory and they could guess what the algorithm is about even before completely going through the algorithm.

3. **Did you find any similarities in the algorithms?** Almost all of the participants were able to determine that the first and last algorithms are identical, with the exception that one algorithm works with an integer array and the other with a string array. One participant was unable to find the similarities but was able to recall solving similar algorithms when both the algorithms were shown side by side.

4. **Did it make a difference to you if the algorithm used integers or strings?** Eleven participants said that they were concentrated on the logic and the control flow of the algorithm they are working with rather than the integer array and the string array. one participant mentioned that it is important to him whether it is an integer or a word that he is working with but the approach he employed to solve the two identical algorithms was same.

5. **Was your approach to solve numbers and words algorithms different?** All the participants said that they used the same approach to solve both numbers and words algorithms as they are identical.

5.3.2 Data Visualization

The data tables presented in the previous section are used to generate the bar chats to visualize the consolidated data. These tables provide information regarding the correctness of answers provided by the participants and also the time take by the participants to respond to the code snippets. This section presents the combination of these tables in the form of bar charts with the results for numbers and words algorithms side by side.

The chart "Total number of correct answers for each numbers and words algorithm" is a combination of "Correctness of Numbers and Words Algorithms Data Table".

The x-axis representing the algorithms, the bar with blue color represents the total number of correct answers provided by the participants for that particular algorithm similarly the orange color bar represents the words algorithm. The y-axis represents the number of correct responses. As it can be seen in the figure 5.4 the algorithm "ArrayAverage" is answered correctly six times, three for numbers algorithm, and three for words algorithm. The rest of the two algorithms are answered eight times correctly, four for numbers algorithm and four for words algorithm.

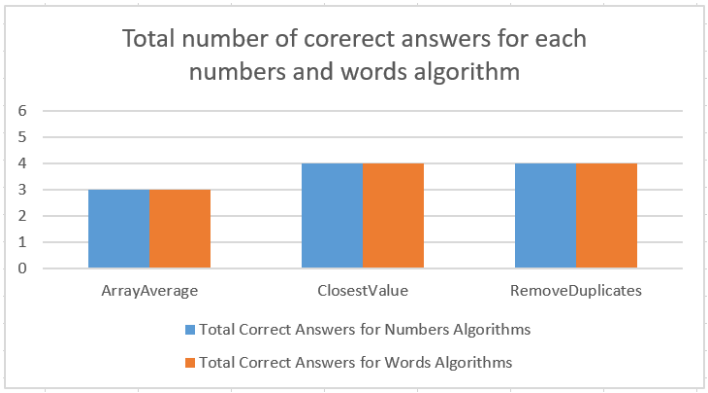


FIGURE 5.4: The bar chart showing total number of correct answers provided by the participants for each words and numbers algorithm

.

The chart "Number of correct answers for numbers Vs. words algorithms" shown in the figure 5.5 is directly derived from the chart shown in the figure 5.4. The y-axis represents the number of responses and the x-axis represents the numbers and words algorithms. This chart represents the total number of correct responses provided by the participants for numbers and words algorithms. Out of twenty-four responses, twenty-one responses are correct, eleven responses of numbers algorithms, and eleven responses of words algorithms.

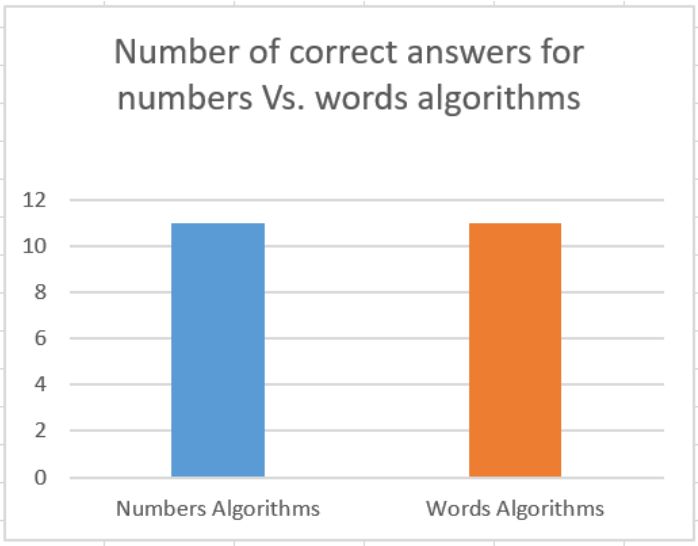


FIGURE 5.5: The bar chat representing the total number of correct answers for all numbers and words algorithms

.

The figure 5.6 is the chart "Average response time for each numbers and words algorithm". This chart is designed to represent the data in the figures 5.2 and 5.3. This chart provides the average time taken by the participant to respond to each numbers and words algorithm. On the x-axis are the algorithms, the blue bar represents the average time taken to respond to numbers algorithm and the orange bar represents the average time taken to respond to the words algorithm. On the y-axis time in seconds is presented. It can be seen in the figure 5.6 that there are slight differences in the response time of about 40 seconds in the "ArrayAverage" algorithm, a difference of about 30 seconds in the "ClosestValue" algorithm, and a negligible difference in the "RemoveDuplicates" algorithm.

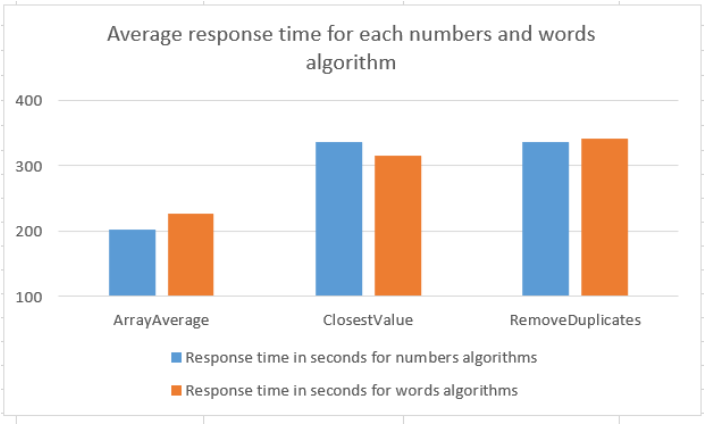


FIGURE 5.6: The chart shows the average time take by the participants to respond to each numbers and words algorithm

.

Figure 5.7 represents the chart "Average response time for numbers Vs. words algorithms". This chart is derived from the chart "Average response time for each numbers and words algorithm". The chart in the figure 5.6 present the average time to solve each algorithm whereas the chart "Average response time for numbers Vs. words algorithms" presents the total average time for numbers and words algorithms. The blue bar represents the average response time for numbers algorithms and the orange bar represents the average response time for words algorithms. The numbers on the y-axis represent time in seconds. From the chart, it can be observed that there is a negligible difference in the response time for numbers and words algorithms.

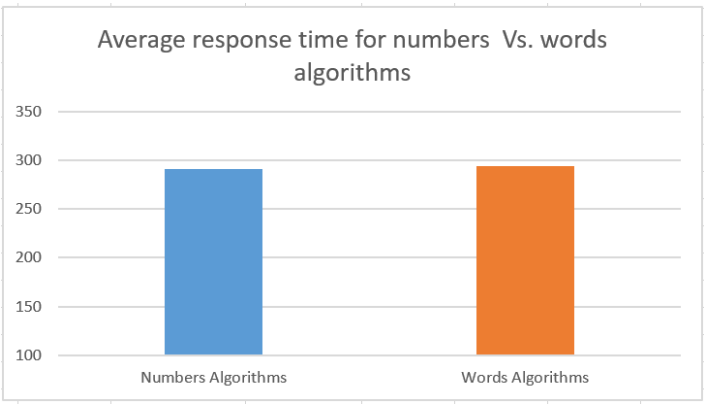


FIGURE 5.7: The bar chart presenting the average response time for all the numbers algorithms Verses words algorithms

.

Chapter 6

Discussion

In this chapter, the summary of the results and the important points of discussion that arose during the implementation and the data collection phase of this study are presented.

The data presented in the previous chapter showed that there is no difference in the correctness of answers provided by the participants to the numbers and words algorithms with identical programming logic. A total of twenty-four questions were answered by participants in a survey, each participant has to answer a questionnaire that presents a numbers algorithm and a words algorithm of similar logic. So out of twelve answers to the numbers algorithms, eleven answers are correct and this corresponds to 91.6 percent. Similarly, for words algorithms eleven out of twelve responses were correct. These results can be seen in the figures 5.5 and 5.4. The Mann-Whitney U test used to test the significance shows that there is no difference in the correctness for numbers and words algorithms.

The data in the previous chapter suggested that there is no difference between numbers and words algorithms with identical programming logic in terms of their response time. The Mann-Whitney U test also indicates the same, there is no difference in the response time for numbers and words algorithms.

The results for the interview conducted at the end of the survey and the transcripts from the participants audio recordings as part of the think-aloud protocol also showed no difference in participants approach in solving the words and numbers algorithms.

The data from the transcripts indicate that the participants are more focused on understanding the snippets with the help of beacons present in the snippets. All the participants followed the same way to understand the numbers and words algorithms.

A think-aloud protocol is used to understand the participants approach to solve the algorithms, though it served as an important tool to observe participants behavior during comprehension and the transcripts generated as part of this was useful conclude about participants approach, it would be difficult to say whether there are any differences in other cognitive processes of participants for numbers and words algorithms. An exception case is observed in the data set, out of twelve participants, one stated that there is a difference in thought process while working with words and numbers, while working with words an extra calculation had to be performed like finding the length of the string. Other participants mentioned as the algorithms are identical, there are no differences in their thought process for words and numbers.

The snippets are present to the participants in a random manner, the participants took more time to respond to the first snippet than the last snippet, this might be because participants getting adjusted to the survey environment, to overcome such effects on the collected data six participants are presented with numbers algorithm at the beginning of the questionnaire followed by the words algorithm. The other six participants are presented with words algorithm first and numbers algorithm next.

6.1 Threats to validity

Threats validity are concerns that must be addressed in the study in order to justify any inaccuracies found. The following are the threats to validity:

- The survey was sent to a group of Computer Science students with varying levels of programming experience, making it difficult to assess the accurate skills of those who responded. As a result, internal validity might get afflicted.

- The participant's demographic and experience data was not particularly important for this study; it was merely collected to observe the distribution of data. The external validity could be affected as a result of this.
- The responses of the three participants are not taken into account for the assessment in order to prevent mistakes and inaccuracies caused by the participants' carelessness. Internal validity could be harmed as a result of this issue.
- There is a slight difference between the numbers and words algorithms the words algorithms have an extra method to find the length of a string. This difference might affect the internal validity.
- Given the nature of the study, the small data set collected did not appear to be sufficient to generate conclusive evidence. But it was adequate to test results against research questions and hypotheses.
- The participants may not say all about what they are thinking while comprehending the numbers and words algorithms this may harm the internal validity.
- Participants are asked to express themselves verbally during the survey. Some of the participants needed to be told that they needed to express themselves clearly and loudly. These interruptions can have a minor effect on the participants thinking processes.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Program comprehension is a broad topic that encompasses a wide range of possibilities for research. The main focus of program comprehension has been on a better understanding of programmers. A variety of objectives were studied by applying various research approaches and strategies over the past three decades. This study is an attempt to understand the behavior of programmers for different programming setups. The objective of this thesis is to study the differences in programmers behavior while they are comprehending the numbers and words algorithms. The research question posed in section 1.2 is answered based on the data obtained from a random set of participants with a different level of experience in the field of Computer Science.

1. Does the correctness of programmers differ for numbers and words algorithms?

   The result set and the statistical test results show that there is no difference in the correctness of programmers for numbers and words algorithms.

2. Does the response time of programmers differ numbers and words algorithms?

   No difference in the participants response time for numbers and words algorithms is observed in the result set and the statistical test results.

Thus the study can be concluded by stating that the study's findings show no difference in programmers behavior for identical numbers and words algorithms.

7.2 Future work

This chapter provides an overview of the future work that can be carried out related to this study. In this thesis work, the difference in comprehension of numbers and words programs is explored. The difference in terms of correctness and response time are studied in this thesis work for this online survey and the think-aloud protocol is used to gather responses from the programmers. This experiment served well for the intended research, in future research on this topic there is a scope for the use of technology to better understand the difference in comprehension of numbers and words programs. With the inclusion of the new technology in a similar experimental setup precise results can be expected. Along with the technology, the study can be conducted with more participants, for a longer duration, and with large programs to produce large result data sets. As a result, the research hypotheses can be backed up by a large number of responses.

The following are some of the suggestions for future research:

- Eye-tracking technology can be used for the same experiment design with more participants to produce more accurate and conclusive evidence. The reading patterns of programmers for numbers and words algorithms can also be accurately recorded with the help of eye-tracking.
- A combination of eye-tracking and fMRI can be used to study the differences in the cognitive process of programmers for numbers and words algorithms.
- "Recall" program comprehension measure can be used with the same type of question asked in the survey to find the differences in programmers behavior for words and numbers algorithms.

Bibliography

[1] Siegmund, Janet and Jana Schumann. "Confounding parameters on program comprehension: a literature survey". In: *Empirical Software Engineering* 20.4 (2015), pp. 1159–1192. ISSN: 1573-7616.

[2] Peitek, Norman, et al., ed. *Simultaneous measurement of program comprehension with fmri and eye tracking: A case study*. 2018.

[3] Siegmund, Janet, ed. *Program comprehension: Past, present, and future*. IEEE, 2016. ISBN: 1509018557.

[4] Siegmund, Janet, et al., ed. *Understanding understanding source code with functional magnetic resonance imaging*. 2014.

[5] M-A. Storey, ed. *Theories, methods and tools in program comprehension: past, present and future*. IEEE, 2005. ISBN: 0769522548.

[6] Amela Karahasanović et al. "Comparing of feedback-collection and think-aloud methods in program comprehension studies". In: *Behaviour & Information Technology* 28.2 (2009), pp. 139–164. ISSN: 0144-929X.

[7] Biggerstaff, Ted J., Bharat G. Mitbander and Dallas Webster., eds. *The concept assignment problem in program understanding*. IEEE, 1993. ISBN: 0818637803.

[8] Matúš. Sulír, ed. *Program comprehension: A short literature review*. 2015.

[9] Siegmund, Janet et al. "Experience from measuring program comprehension-toward a general framework". In: *Software Engineering 2013* (2013). ISSN: 38857960.

[10] Ben Shneiderman. "Exploratory experiments in programmer behavior". In: *International Journal of Computer & Information Sciences* 5.2 (1976), pp. 123–143. ISSN: 1573-7640.

[11] Nancy Pennington. "Stimulus structures and mental representations in expert comprehension of computer programs". In: *Cognitive psychology* 19.3 (1987), pp. 295–341. ISSN: 0010-0285.

[12] Ben Shneiderman. "Human factors in computer and information systems". In: *Cambridge, MA: Winthrop* (1980).

[13] Elliot Soloway and Kate Ehrlich. "Empirical studies of programming knowledge". In: *IEEE Transactions on Software Engineering* 5 (1984), pp. 595–609. ISSN: 0098-5589.

[14] Alastair Dunsmore and Marc Roper. "A comparative evaluation of program comprehension measures". In: *The Journal of Systems and Software* 52.3 (2000), pp. 121–129.

[15] Wilhelm Max Wundt. *Grundzüge der physiologischen Psychologie*. W. Engelman, 1874.

[16] John R. Anderson. "Methodologies for studying human knowledge". In: *Behavioral and Brain Sciences* 10.3 (1987), pp. 467–477. ISSN: 1469-1825.

[17] Stanley Letovsky. "Cognitive processes in program comprehension". In: *Journal of Systems and software* 7.4 (1987), pp. 325–339. ISSN: 0164-1212.

[18] Anneliese von Mayrhauser and Stephen Lang. "A coding scheme to support systematic analysis of software comprehension". In: *IEEE Transactions on Software Engineering* 25.4 (1999), pp. 526–540. ISSN: 0098-5589.

[19] Teresa M. Shaft and Iris Vessey. "The relevance of application domain knowledge: The case of computer program comprehension". In: *Information systems research* 6.3 (1995), pp. 286–299. ISSN: 1047-7047.

[20] Anneliese von Mayrhauser and A. Marie Vans, eds. *From program comprehension to tool requirements for an industrial environment*. IEEE, 1993. ISBN: 0818640421.

[21] Janet Hughes and Steve Parkes. "Trends in the use of verbal protocol analysis in software engineering research". In: *Behaviour & Information Technology* 22.2 (2003), pp. 127–140. ISSN: 0144-929X.

[22] David C. Littman et al. "Mental models and software maintenance". In: *Journal of Systems and software* 7.4 (1987), pp. 341–355. ISSN: 0164-1212.

[23] Ben Shneiderman. "Measuring computer program quality and comprehension". In: *International journal of man-machine studies* 9.4 (1977), pp. 465–478. ISSN: 0020-7373.

[24] Robert S. Rist, ed. *Plans in programming: definition, demonstration, and development*. 1986.

[25] Riston Tapp and Rick Kazman, eds. *Determining the usefulness of colour and fonts in a programming task*. IEEE, 1994. ISBN: 0818656476.

[26] Paul W. Oman and Curtis R. Cook. "Typographic style is more than cosmetic". In: *Communications of the ACM* 33.5 (1990), pp. 506–520. ISSN: 0001-0782.

[27] Siegmund, Janet. "Framework for measuring program comprehension". In: (2012).

[28] Anneliese von Mayrhauser and A. Marie Vans. "Program comprehension during software maintenance and evolution". In: *Computer* 28.8 (1995), pp. 44–55. ISSN: 0018-9162.

[29] Elliot Soloway, Beth Adelson, and Kate Ehrlich. "Knowledge and processes in the comprehension of computer programs". In: *The nature of expertise* (1988), pp. 129–152.

[30] Anneliese von Mayrhauser and A. Marie Vans., eds. *Comprehension processes during large scale maintenance*. IEEE, 1994. ISBN: 081865855X.

[31] Ben Shneiderman and Richard Mayer. "Syntactic/semantic interactions in programmer behavior: A model and experimental results". In: *International Journal of Computer & Information Sciences* 8.3 (1979), pp. 219–238. ISSN: 1573-7640.

[32] Ruven Brooks. "Towards a theory of the comprehension of computer programs". In: *International journal of man-machine studies* 18.6 (1983), pp. 543–554. ISSN: 0020-7373.

[33] J. W. Belliveau et al. "Functional mapping of the human visual cortex by magnetic resonance imaging". In: *Science* 254.5032 (1991), pp. 716–719. ISSN: 0036-8075.

[34] Korbinian Brodmann. *Brodmann's: Localisation in the cerebral cortex*. Springer Science & Business Media, 2007. ISBN: 0387269193.

[35] Siegmund, Janet, et al., ed. *Measuring neural efficiency of program comprehension*. 2017.

[36] Keith Rayner. "Eye movements in reading and information processing". In: *Psychological bulletin* 85.3 (1978), p. 618. ISSN: 1939-1455.

[37] Päivi Majaranta and Andreas Bulling. "Eye tracking and eye-based human–computer interaction". In: *Advances in physiological computing*. Springer, 2014, pp. 39–65.

[38] Kenneth Holmqvist et al. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011. ISBN: 0191625426.

[39] Busjahn, Teresa, et al, ed. *Eye movements in code reading: Relaxing the linear order*. IEEE, 2015. ISBN: 1467381594.

[40] Sharif, Bonita, and Huzefa Kagdi., ed. *On the use of eye tracking in software traceability*. 2011.

[41] Dag I. K. Sjøberg et al. "A survey of controlled experiments in software engineering". In: *IEEE Transactions on Software Engineering* 31.9 (2005), pp. 733–753. ISSN: 0098-5589.

[42] Sharafi, Zohreh, et al., ed. *Eye-tracking metrics in software engineering*. IEEE, 2015. ISBN: 1467396443.

[43] Martha E. Crosby and Jan Stelovsky. "How do we read algorithms? A case study". In: *Computer* 23.1 (1990), pp. 25–35. ISSN: 0018-9162.

[44] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. "A systematic literature review on the usage of eye-tracking in software engineering". In: *Information and Software Technology* 67 (2015), pp. 79–107. ISSN: 0950-5849.

[45] Unaizah Obaidellah, Mohammed Al Haek, and Peter C-H Cheng. "A survey on the usage of eye-tracking in computer programming". In: *ACM Computing Surveys (CSUR)* 51.1 (2018), pp. 1–58. ISSN: 0360-0300.

[46] Crosby, Martha E., Jean Scholtz, and Susan Wiedenbeck., ed. *The Roles Beacons Play in Comprehension for Novice and Expert Programmers*. Citeseer, 2002.

[47] Bednarik, Roman, and Markku Tukiainen., ed. *An eye-tracking methodology for characterizing program comprehension processes*. 2006.

[48] Busjahn, Teresa, Carsten Schulte, and Andreas Busjahn., ed. *Analysis of code reading to gain more insight in program comprehension*. 2011.

[49] Sharafi, Zohreh, et al., ed. *Women and men—different but equal: On the impact of identifier style on source code reading*. IEEE, 2012. ISBN: 1467312169.

[50] Busjahn, Teresa, Roman Bednarik, and Carsten Schulte., ed. *What influences dwell time during source code reading? Analysis of element type and frequency as factors*. 2014.

[51] Peitek, Norman, et al. "A look into programmers' heads". In: *IEEE Transactions on Software Engineering* 46.4 (2018), pp. 442–462. ISSN: 0098-5589.

[52] Duraes, João, et al., ed. *WAP: understanding the brain at software debugging*. IEEE, 2016. ISBN: 146739002X.

[53] Fritz, Thomas, et al., ed. *Using psycho-physiological measures to assess task difficulty in software development*. 2014.

[54] Fakhoury, Sarah, et al, ed. *The effect of poor source code lexicon and readability on developers' cognitive load*. IEEE, 2018. ISBN: 1450357148.

[55] T. Kluthe. "A measurement of programming language comprehension using p-BCI: An empirical study on phasic changes in alpha and theta brain Waves". In: *Master's thesis, Southern Illinois University Edwardsville, Edwardsville, IL, USA* (2014).

[56] B. Chance et al. "Cognition-activated low-frequency modulation of light absorption in human brain". In: *Proceedings of the National Academy of Sciences* 90.8 (1993), pp. 3770–3774. ISSN: 0027-8424.

[57] *SoSci Survey the Professional Solution for Your Online Survey*. https://www.soscisurvey.de/. (Accessed on 04/24/2021).

[58] *Open Broadcaster Software | OBS*. https://obsproject.com/. (Accessed on 04/24/2021).

[59] *Mann–Whitney U test - Wikipedia*. https://en.wikipedia.org/wiki/Mann-Whitney_U_test. (Accessed on 04/24/2021).

[60] *(7) How To... Perform the Mann-Whitney U Test (By Hand) - YouTube*. https://www.youtube.com/watch?v=BT1FKd1Qzjw&ab_channel=EugeneO. (Accessed on 04/24/2021).

[61] *Otter Voice Meeting Notes - Otter.ai*. https://otter.ai/. (Accessed on 04/24/2021).

[62] *Understanding Hypothesis Tests: Significance Levels (Alpha) and P values in Statistics*. https://blog.minitab.com/en/adventures-in-statistics-2/understanding-hypothesis-tests-significance-levels-alpha-and-p-values-in-statistics#:~:text=Thesignificancelevel-alsodenoted,thereisnoactualdifference.. (Accessed on 04/24/2021).

Appendix A

Questionnaire

This section presents a completed sample questionnaire used to gather the responses from the participants. All the snippets used in this study are provided in the CD attached to this thesis report. The transcripts that are generated from the audio recordings are provided in the remote repository. Here is the link to the repository that contains the transcripts generated as part of think-aloud study "https://github.com/Anirudh-Adavalli/Think-Aloud$_{S}$tudy$_{T}$ranscript".

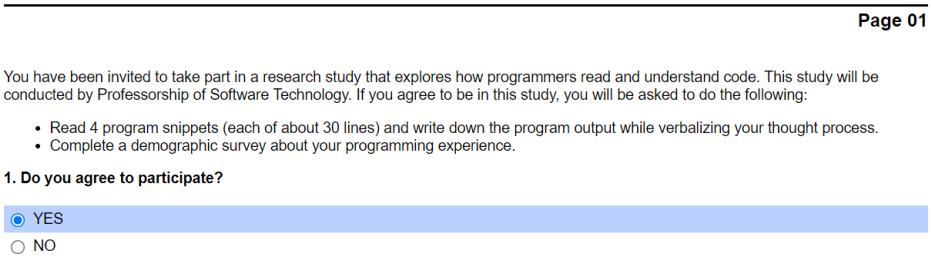


FIGURE A.1: This is the first page in the questionnaire asking for the participants consent

.

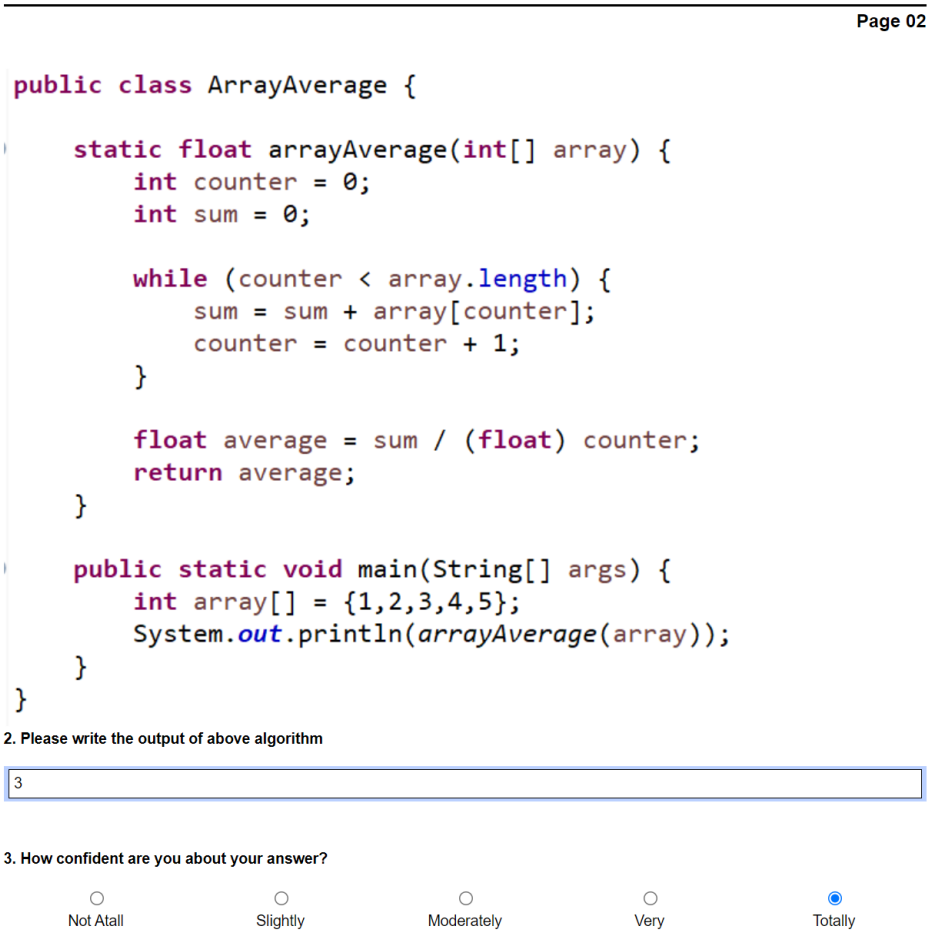


FIGURE A.2: This is the first comprehension task and is a numbers algorithm

.

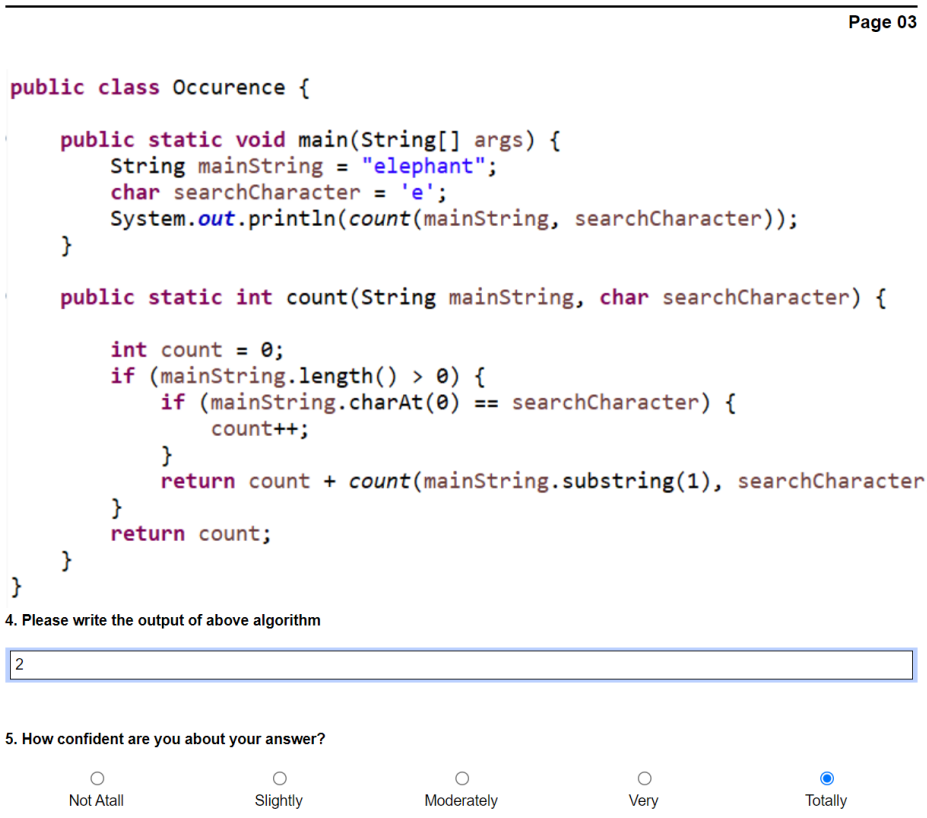


FIGURE A.3: Comprehension task two: This is an algorithm to distract the participants

.

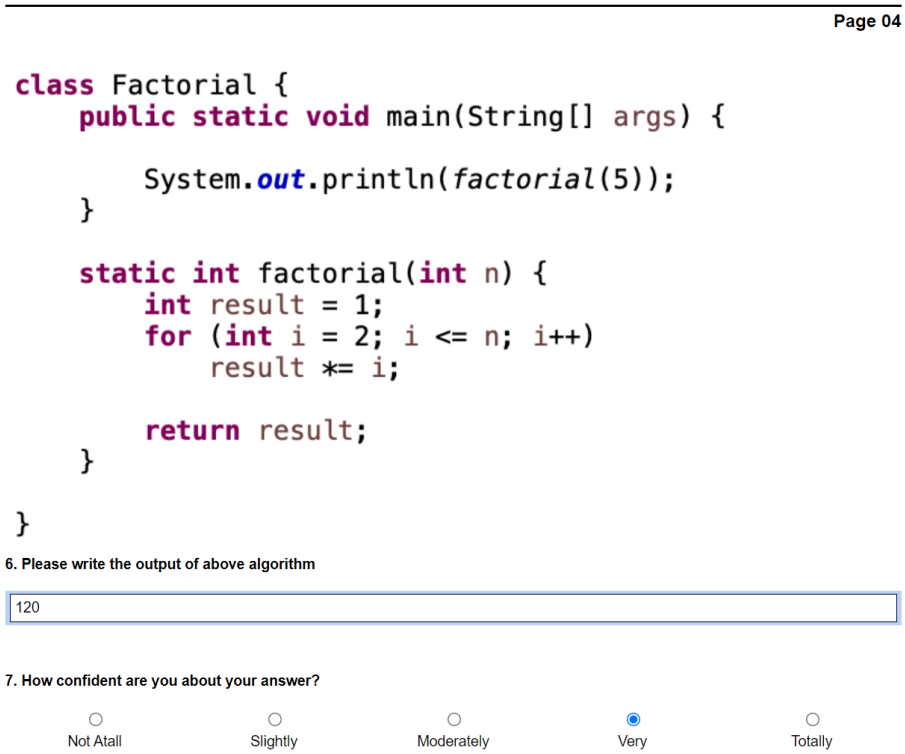


FIGURE A.4: Comprehension task three: This is an algorithm to distract the participants

.

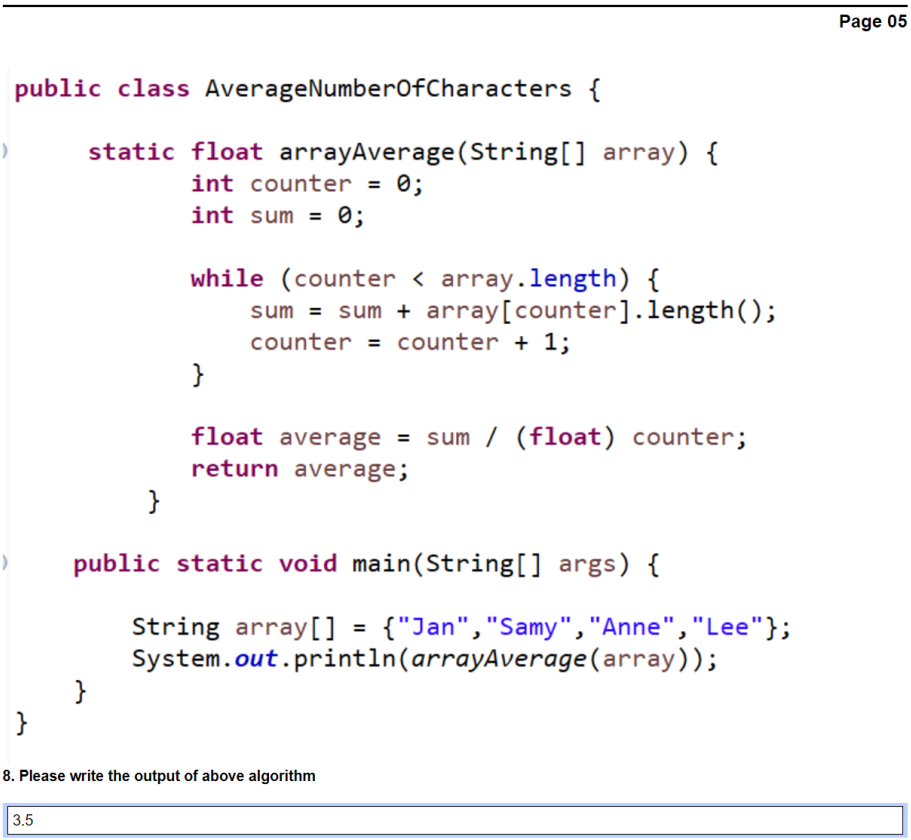


FIGURE A.5: This is the last comprehension task in the questionnaire, a words algorithm identical to the algorithm in the first task.

.

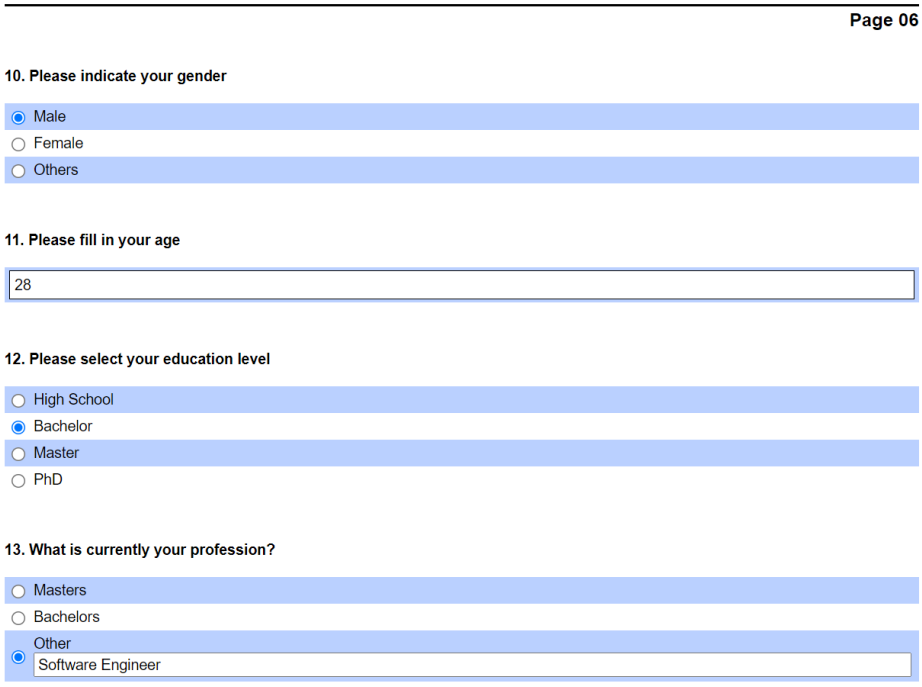
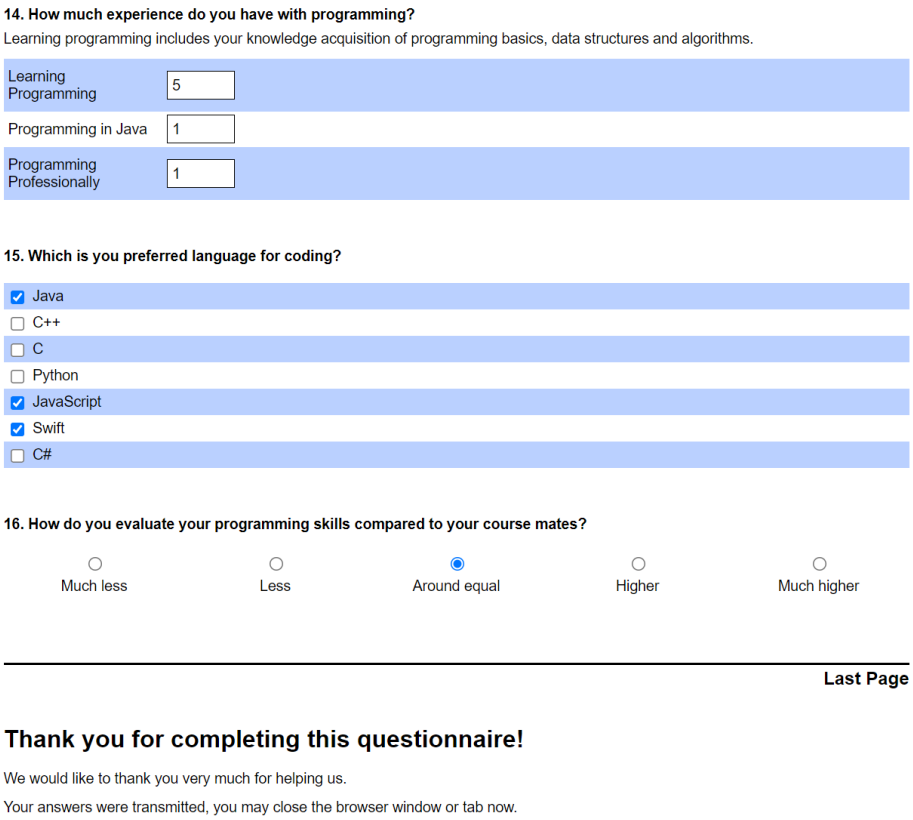


FIGURE A.6: Demographic questions

.



14. How much experience do you have with programming?

Learning programming includes your knowledge acquisition of programming basics, data structures and algorithms.

Learning Programming	5
Programming in Java	1
Programming Professionally	1

15. Which is you preferred language for coding?

- [x] Java
- [ ] C++
- [ ] C
- [ ] Python
- [x] JavaScript
- [x] Swift
- [ ] C#

16. How do you evaluate your programming skills compared to your course mates?

Much less | Less | Around equal | Higher | Much higher

Last Page

Thank you for completing this questionnaire!

We would like to thank you very much for helping us.

Your answers were transmitted, you may close the browser window or tab now.

FIGURE A.7: Experience and Skill questions

.