Bachelor's Thesis

EXPLORATION OF SOFTWARE TOOLS USED FOR ANALYZING EYE TRACKING DATA IN SOFTWARE ENGINEERING

TIMON DÖRZAPF

October 10, 2023

Advisor:

Marvin Wyrich	Chair of Software Engineering
Dr. Norman Peitek	Chair of Software Engineering

Examiners:

Prof. Dr. Sven Apel	Chair of Software Engineering
Prof. Dr. Anna Maria Feit	Chair of Human-Computer Interaction

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,______________________ ______________________________

(Datum/Date) (Unterschrift/Signature)

ABSTRACT

A lot of empirical studies in software engineering use eye tracking to expand their data points. Yet, there is still no standardization of the tools that are used to analyze eye tracking data.

This poses a problem as comparability between studies that used different eye tracking software tools is difficult. The results of a study may be influenced by the choice of the eye tracking software tool to analyze obtained study data.

We explored and evaluated the software tools used by performing a lightweight Systematic Mapping Study (SMS) and extracted the eye tracking software tools used from the papers found. We then compared eye tracking metrics calculated by different software tools. We found that there are significant differences in the results of software tools for the same data set. This finding may impact the comparability and reproducibility of all studies that used eye tracking, especially those that did not name the software tool they used. Additionally, our evaluation of the current state of the art of software tools used in eye tracking studies in software engineering enables future work to know which software tools may be the best fit for their study.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOI	Area Of Interest
AOG	Area Of Glance
EEG	Electroencephalogram
fMRI	functional Magnetic Resonance Imaging
GSV	Galvanic Skin Response
IDE	Integrated Development Environment
RFV	Restricted Focus Viewer
SMS	Systematic Mapping Study
SLR	Systematic Literature Review

1 INTRODUCTION

Eye tracking in software engineering research has gotten more and more popular in recent times. An argument as to why this is happening is the reduced costs of the hardware, as eye tracking devices were rather expensive some decades ago compared to the prices of today where low end ones start at 100€ [19]. Another advantage is that it helps studies to understand the thought processes of developers by recording significant evidence about how participants interact with visual information [19]. It has been used in a wide range of applications such as code comprehension, debugging and traceability [20, 21]. In combination with other psycho-physiological measures, an even deeper understanding of the way developers work can be achieved as eye tracking complements the other measures well. It can for example be used to know which part of a snippet of code a participant is looking at when using functional Magnetic Resonance Imaging (fMRI), such that the data of the fMRI can be evaluated in a more efficient manner.

Even though eye tracking has been used for more than 30 years, there still is not a standard way of performing and analyzing eye tracking experiments [18–20]. Particularly, there is a large variety of tools that are used to analyze the eye tracking data [19, 20]. Due to the wide amount of tools, it is difficult to choose the correct one for the requirements of a study, as not every tool is compatible with every eye tracker. The tools also have different capabilities. Depending on the needs of the study, it is important to choose the right tool that has the required capabilities.

Some of these tools are closed-source and only work for the brand of eye trackers they have been developed for. Additionally, the uncertainty of which code is behind the tool makes them a weaker choice. Another point is that it is not possible to know the algorithms behind the calculations, as you can not compare the calculations that were made if the used algorithm is not known. All in all, this makes comparisons between studies a difficult objective when using closed-source tools.

But compatibility between open-source tools is also a point that has not been discussed in detail in literature. Even though the code and the algorithms that are used are publicly available, there have not been studies which measure if the results from these tools are equivalent or not. Thus also studies that use open-source tools are difficult to compare.

Goal of this Thesis

The goal of this thesis is to explore and evaluate the tools that were used during eye tracking experiments in software engineering in a systematic way. This way, we were able to see which tools have been used and how the usage of tools has evolved. Additionally, the effects of using different tools for the same data was measured in an explorative way.

Another goal of this research is to lead to a deeper understanding of the question if using different tools will lead to different results. By evaluating the usage of the different tools and comparing the results of different tools with the same data, it can be seen if using a different tool during the data analysis part has a significant effect on the result of the study. This in turn will lead to knowing if one can use different tools without consequences or if studies should use a standardized tool for the data analysis to make studies comparable, which in turn will make it easier to perform eye tracking experiments. As there was no such a systematic literature research conducted so far, the results of this thesis will advance the research of eye tracking studies in software engineering by providing a deeper insight into the tools used in the data analysis part of the studies and may help reducing threats to validity introduced by using certain eye tracking software tools.

Overview

After this introduction, we first cover underlying concepts about eye tracking in Chapter 2. We give an overview about the theory and assumptions of eye tracking. Then we will present how eye tracking devices function. Finally, we explain some visual effort metrics which are used to analyze eye tracking data.

Following this chapter, we will present some related work in Chapter 3. We cover papers that provide additional information to eye tracking in software engineering and software tools that were specifically developed to be used with eye tracking.

Afterwards, we describe our methodology in Chapter 4. There we present our research questions. We present how we found the papers for our exploration of eye tracking software tools. Then we describe how we performed the data analysis part of the thesis.

In Chapter 5, we present the results of our thesis. We also discuss those results and their implications.

Additionally, we provide the threats to validity of our thesis and how we tried to minimize those in Chapter 6. Finally, in Chapter 7, we summarize our findings and provide a conclusion while also giving an outlook on further ideas to this subject.

2 BACKGROUND

This chapter will provide necessary background information for this thesis.

2.1 EYE TRACKING THEORY

Eye tracking devices are used to collect the eye movements of a person such that it can be seen where the person is looking at. This makes it possible to monitor a person's visual attention when looking at a stimulus when working on a task, as the eye movement is essential to cognitive processes of the brain when performing software engineering tasks. A stimulus is the object that is needed to perform the task, for example the source code. A task can be such thing as program comprehension of source code. The resulting data from eye trackers is analyzed with respect to certain areas called Area Of Interest (AOI). These AOI can be relevant or not to a person during a task. If the task is source code comprehension, a relevant AOI could be a function name whereas an irrelevant AOI could be a comment about something different. Another metric is the Area Of Glance (AOG). These can be either the whole stimulus or another AOI.

Eye gaze data, which is obtained by processing the raw movement data, can be classified into the following categories [5]:

- Saccade: rapid eye movements that are used to reposition the fovea to a new location in the visual environment. They range in duration from 10 to 100 ms, which renders the person effectively blind during the transition. They occur between fixations.
- Fixation: eye movement that stabilizes the retina over a stationary object of interest. They range in duration from 150 to 600 ms, but this changes depending on the task and the characteristics of the person.
- Scan path: a series of fixations or visited AOIs in a chronological order that appear due to the eyes fixating on different parts of a stimulus.
- Pupil dilation and constriction: the dilation of the pupil, the aperture through which the light enters the eye, is controlled by the iris muscle. According to Poole and Ball [15], a larger pupil indicates increased cognitive effort.

The following assumptions about eye tracking have been made [9]:

- The immediacy assumption: states that a person tries to interpret a stimulus as soon as the person sees it.
- The eye-mind assumption: states that until a person understands a stimulus, they fixate their attention on it.

These assumptions build the foundation of how eye gaze data represents the person's cognitive processes. Sharafi et al [20] performed a Systematic Literature Review on eye tracking studies in software engineering, which was used as a basis for RQ1 of this thesis. The papers that were analyzed by Sharafi et al [20] support these assumptions and also used them to analyze and interpret the eye-movement data. Another assumption of these papers is that the person is actively engaged in performing their task.

2.2 EYE TRACKING DEVICES

There is a wider variety of eye trackers available for scientific and for business purposes [20]. Depending on the eye tracker, they may have a different form or use different methods to track eye gaze. An eye tracker normally consists of the following hardware and software components [19]:

- One or more (usually infrared) cameras
- One or more (usually infrared) light sources
- Image-processing software that detects and locates the eyes and the pupils and maps eye motion and the stimulus.
- Data collection software to collect and store real-time eye gaze data.
- Real-time display showing the location of the eyes' focus.

Currently, most available eye trackers use the corneal-reflection/pupil-center method where an emitter of (usually infrared) light is directed toward the eyes, entering the pupils [19]. A small amount of light is reflected by the eyes whereas a significant amount of light is reflected back in the pupils which causes the pupils to appear bright. The eye tracking cameras can detect those reflections and therefore track the movements of the eye.

2.3 VISUAL EFFORT METRICS

In this section we will present some visual effort metrics that are used to measure the visual effort that is representative of the task and stimuli being assessed [18].

Metrics based on Fixations

- Fixation Count (FC): total number of fixations in each AOI. According to Goldberg et al [6], a higher number of fixations on a specific stimulus shows that the search for finding relevant information is not efficient.
- Fixation Rate (FR): Can be calculated by the following formula:

$$FR = \frac{\text{Total number of Fixations in AOI}}{\text{Total Number of fixations in AOG}}$$

  These can be either the whole stimulus or another AOI. The fixation rate shows the ratio of fixations between two different AOIs. If FR is smaller, the search task is less efficient. If FR is higher and we are looking at comprehension tasks, then the person shows great interest in an AOI or the AOI is difficult to understand.

- Fixation Spatial Density (SD): If we consider the stimulus as a grid, then SD is equal to the number of cells that contain at least one fixation divided by the total number of cells. It represents the coverage of an area and gives a measure of the dispersion of the person's fixations. It can be calculated as follows:

  $$SD = \frac{\sum_{i=1}^{n} c_i}{n}$$

  with n the number of cells in the grid. If the cell of number i is visited, c_i is 1 else 0.

- Convex Hull Area: the area of the smallest convex set of fixations which contain all fixations of a person. It visualizes the spatial distribution of fixations and shows the preferred parts of a visual stimulus.

Metrics based on the Duration of Fixations

- Average Fixation Time (AFD): sum of durations of all fixations divided by the number of fixations.

  $$AFD(AOI) = \frac{\sum_{i=1}^{n}(ET(F_i) - ST(F_i)) \text{ in AOI}}{n}$$

  with $ET(F_i)$ the end time, $ST(F_i)$ the start time for a fixation F_i. n is the total number of fixations in a given AOI.

- Ratio of ON-target:All-target Fixation Time (ROAFT): sum of the durations of all fixations in a AOI divided by the total duration of all fixations for the AOG. Lower efficiency is indicated by a smaller ratio.

  $$ROAFT = \frac{\sum_{i=1}^{n}(ET(F_i) - ST(F_i)) \text{ in AOI}}{\sum_{j=1}^{n}(ET(F_j) - ST(F_j)) \text{ in AOG}}$$

- Fixation Time (FT): sum of the durations of all fixations in an AOI.

- Average Duration of Relevant Fixations (ADRF): total duration of the fixations for relevant AOIs.

  $$ADRF = \frac{\text{Fixations Duration of Relevant AOIs}}{\text{Total Number of Relevant AOIs}}$$

- Normalized Rate of Relevant Fixations (NRRF): compares two or more stimuli with each other. A stimuli requires more visual effort if it requires more relevant fixations.

  $$NRRF = \frac{\text{ADRF}}{\left(\frac{\text{Fixation Duration of all AOIs}}{\text{Number of all AOIs}}\right)}$$

  NRRF is normalized to adjust for the size of stimulus using the total number of AOIs in the stimulus.

Metrics based on Saccades

- Number of Saccades and Saccade Duration: identical to their corresponding fixation-based metrics.
- Regressions Rate: percentage of backwards saccades of any length. The higher the percentage, the better can the person read and understand the stimulus.

Metrics based on Scanpaths

- Attention Switching Frequency: dynamics of visual attention using the total number of switches between a set of AOIs per minute.
- Transitional Matrix: tabular representation of frequencies of transitions between AOIs. Higher density indicates extensive search with inefficient scanning on a stimulus whereas a sparse matrix indicates a directed and efficient search.

$$TM = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}}{n^2}$$

- ScanMatch: compares scanpaths and computes a similarity score. It is based on the Needleman-Wunsch algorithm.
- Linearity: search strategy of a person for a stimulus. It uses eye-movements linearity to characterize how a person read source code.

2.4 FIXATION AND SACCADE ALGORITHMS

A lot of different fixation algorithms exist today [1]. In this section, we will present some of them:

IVT Algorithm

A widely employed approach for distinguishing between samples associated with fixations and those linked to saccades is to analyze their velocities. The Identification by Velocity Threshold (IVT) algorithm, as outlined by Salvucci and Goldberg [16], operates on this principle. It employs a predefined velocity threshold to classify data into fixations and saccades. In this context, fixations encompass segments of data with point-to-point velocities below the specified threshold, while saccades include segments with velocities exceeding this threshold.

This fundamental velocity-based criterion often serves as the foundation for various other algorithms. The specific implementation discussed here is attributed to Komogortsev et al. [11].

IDT Algorithm

One of the most widely used algorithms for detecting fixations is the Identification by Dispersion-Threshold (IDT) algorithm. This algorithm, as outlined by Salvucci and Goldberg [16], is rooted in the data reduction technique developed by Widdel [22]. The IDT algorithm operates on both the x and y data coordinates and relies on two predefined thresholds: the maximum fixation dispersion threshold and the minimum fixation duration threshold.

In order for a segment of data to be classified as a fixation, it must meet the minimum duration threshold, which means that data samples covering a duration equal to or longer than this threshold must fall within a spatial area not exceeding the dispersion threshold. Any data samples that satisfy these criteria are designated as belonging to a fixation.

IMST Algorithm

Another approach to event detection involves the use of the Identification by Minimal Spanning Tree (IMST) algorithm. This algorithm constructs a “tree” representation of the data, with branches extending to individual data samples. The primary goal of the algorithm is to create a tree structure that minimizes branching, ensuring that samples from distinct clusters are associated with separate nodes higher up in the tree, rather than being forced into an extensive branching structure linked to a single node at a lower level.

By imposing specific thresholds on the samples at the edges of a cluster, the IMST algorithm can effectively identify saccades and exclude them from the fixation detection process. This particular implementation of the IMST algorithm is credited to Komogortsev et al. [11].

3 RELATED WORK

Sharafi et al [20] performed a Systematic Literature Review (SLR) in 2015 on the usage of eye tracking in software engineering. They analyzed 36 papers between 1990 and 2014 using SLR to evaluate the current state of art of eye tracking. A limitation of current eye tracking studies that was found was that several tools were used to analyze eye tracking data. The authors named Taupe and OGAMA as examples for open-source software that is working with many commercial eye trackers.

In “A Practical Guide on Conducting Eye Tracking Studies in Software Engineering” [19], the authors also commented on the fact that due to a lack of standardized protocols and tools, comparisons across studies are difficult. Commercial eye trackers suppliers are providing their closed source data analysis tools, but there are open-source alternatives which provide the ability to work with most commercial eye trackers. They contribute a list of 6 open-source software data analysis tools which they would recommend based on their experience.

In recent years, there has been effort made to create more open-source software to analyze eye tracking data [4, 7, 23]. These tools can be utilized with frequently used commercial eye trackers by software engineering researchers, which makes them a good alternative to the paid closed-source software of commercial eye tracker suppliers. Zyrianov et al [23] presents Deja Vu, which can be used to record eye tracking and all telemetry data such that high-speed, high-quality eye trackers can overcome their real-time limitations of mapping screen coordinates to lines and columns, which in turn facilitates code comprehension studies and more.

Andersson et al. [1] provided an overview over the currently existing algorithms to parse eye movement data. They tested ten different algorithms on the same data set to provide a comparison between those algorithms and to find out out which algorithms performs the best. Their results were that depending on the algorithm, the resulting event duration had a great variance depending on the algorithm that was used. For static stimuli, fixation and saccade detection were working relatively well, but for all other measures and if dynamic stimuli was used, the algorithms did not provide good results. They also found that the LNS algorithm by Larsson, Nyström & Stridh [12] was the best performing algorithm for saccade detection.

We can see that the lack of a standardized data analysis procedure is a problem that has been known for some time. The increasing number of open-source software tools available for analyzing eye tracking data makes it difficult to choose the right tool for a study. In addition, the algorithms available can have a large impact on the results of the study. Therefore, it is important to know which software tools are used in eye tracking studies in software engineering and how the results of different software tools compare to each other.

4 METHODOLOGY

We will adopt part of the methodology known from conducting SMS to find a dataset of relevant eye tracking studies. A SMS a is structured method that categorizes research reports and results that have been published [14]. As we only want to categorize specific parts of research reports and results, a partial execution of a SMS was chosen since this is well suited for our objective of exploring the software tools used for eye tracking studies. Such a SMS helps in identifying research gaps in eye tracking studies in software engineering [14]. As we only use a partial execution of a SMS, we will not perform keywording using abstracts and a classification scheme as described in [14]. In addition, only extract a subset of the data items that are usually extracted in a SMS as described in [14], as we have no need for the other data items.

4.1 RESEARCH QUESTIONS

As we will perform a part of a SMS on papers that use eye tracking in software engineering studies, it is important to see which tool they used to analyze the data produced by the eye tracker. As there are a lot of different software tools available, which are also categorized into open-source and closed-source software tools, providing a summary of the software tools used and their capabilities is an important insight. Therefore, the first research question is:

RQ1: Which software tools are used to analyze eye tracking data in software engineering studies?

In addition to the software tools used, it is also important to know why the authors chose these software tools. Therefore, the second research question is:

RQ2: How do authors justify their choices of software tools and what consequences do they discuss?

There have been discussions about the subject of comparability between this kind of studies as it is difficult to compare the results of studies if the algorithm used to analyze the data may be different. The comparability between eye tracking software tools is largely unexplored. By comparing different software tools on the same data set, it may be possible to see if certain software tools produce different results than other software tools. As such, the third research question is:

RQ3: Which effects on the results and conclusions does using different software tools on the same experiment data have?

4.2 SEARCH PROCESS

To find papers using eye tracking in software engineering for the part of a SMS, we used the digital libraries of IEEEXplore[1] and ACM[2]. As IEEEXplore and ACM both contain a lot of software engineering specific papers and there is not a lot of overlap between those two libraries [2]. As it was possible to use the same search query for both online libraries, the internal validity of the results was not threatened in this regard. Other libraries such as ScienceDirect or indexers such as Google Scholar were not used as this would have been beyond the feasibility of a bachelor thesis.

The main goal of our study is to find all papers that use eye tracking with eye trackers while performing tasks using software engineering-related stimuli such as code comprehension. Therefore, we define three sets of keywords with the stipulation of having at least one keyword of each set present in the paper:

```
((“eye-track∗” OR “eye track∗” OR "eyetrack∗") AND
(code OR program∗ OR representation∗) AND
(comprehen∗ OR understand∗ OR
debug∗ OR read∗ OR scan∗))
```

These search terms are similar to those of Sharafi et al [20] used for their literature research. Their search string was:

```
((“eye-track∗” OR “eye track∗” OR "RFV" OR "Restricted Focus Viewer") AND
(source code OR program∗ OR "UML" OR "model∗" OR representation∗) AND
(comprehen∗ OR understand∗ OR
debug∗ OR "navigat∗" OR read∗ OR scan∗))
```

These are nearly the same as those of Sharafi et al. [20] except for the additional stipulations that papers that use a Restricted Focus Viewer (RFV) are not allowed and that eye-tracking experiments need to be conducted in front of a screen. I based my search terms on Sharafi et al. [20], as they already conducted a literature search of eye tracking studies in software engineering, which is a superset of the papers that we wanted to find, as it included more varied search terms. Therefore, it was probable that those search terms would yield good results for new papers of the same type.

As Sharafi et al. [20] already conducted a literature search for the years 1990–2014, it was not deemed necessary to replicate this search for the years before 2015 as it would probably lead to the same or fewer results. Thus, the literature search was conducted from 2015 to 2023.

Due to time constraints, no snowballing was performed.

4.3 SELECTION PROCESS

The process to select the papers is the following:

1. Selecting related papers by search engines: The previously described query is executed in the online libraries of IEEEXplore and ACM. The search engines for IEEEXplore

1 https://ieeexplore.ieee.org
2 https://dl.acm.org/

searched the metadata of the papers, which includes the abstract, index terms, and bibliographic citation data (such as document title, publication title, author, etc.). This search produced 173 results.

The search engine for ACM searched the abstract for the search terms. This search produced 136 results.

As the search engine of ACM did not include a metadata option, the abstract search was the best option to receive similar results. In total, the search of the IEEEXplore and ACM libraries provided 309 results.

2. Selecting related papers by previous study: The 36 papers that Sharafi et al. [20] found at the end of their selection process were added to the list of possible related papers.

   In total, there are now 348 possible related papers.

3. Duplicate removal: by manual checking of the papers, duplicates were removed. In total, 16 duplicates were removed such that 332 papers were left.

4. Quick Check: The 348 papers were checked for relevance by reading the title and abstract. If the papers were either not relevant to software engineering or did not conduct an eye tracking study, the papers were removed. After this step, there was a total of 109 papers left.

5. Applying inclusion/exclusion criteria: by applying the inclusion and exclusion criteria which are described in the following section, we checked whether the papers were relevant or not to the goal of this thesis.

   After the criteria were applied, there were 97 papers left, which is the final number of papers found.

Inclusion and Exclusion Criteria

After the papers were collected, there was be a selection process with predefined inclusion and exclusion criteria. I used nearly the same inclusion criteria as Sharafi et al. [20] with two additional exclusion criteria and a slight modification of one exclusion criteria:

- I1: Primary studies published in journals or conference and workshop proceedings in the form of experiments, surveys, case studies, reports, and observation papers using eye-tracking technology to study and investigate software engineering activities.

- I2: Primary studies that present the more detailed and complete results if there is more than one published version of a specific study.

The exclusion criteria are the following:

- E1: Papers that do not use an eye-tracker or that use a RFV.
- E2: Papers that are not related to software engineering.
- E3: Papers in "gray" literature, which are not published by trusted, well-known publishers, and– or which did not go through a well-defined referring process.
- E4: Papers that are not published in English.
- E5: Papers where eye-tracking experiments were not conducted in front of a screen.
- E6: Papers re-reporting the results, or doing a re-analysis of a previously published experiment were studied, and the most comprehensive paper was selected.
- E7: Papers not involving an empirical study or only those that propose a proof of concept.

The slight modification of the exclusion criteria compared to those of Sharafi et al. is that papers that use a RFV are not allowed and that eye-tracking experiments need to be conducted in front of a screen.

The two additional exclusion criteria are E6 and E7, as we only wanted to include studies that created new data from eye tracking, as otherwise there may have been threats to validity by reusing the same dataset.

Choice of Data Set for RQ3

Of the 97 analyzed studies, 28 studies had openly accessible replication packages available. Those were mentioned either in the paper or in the supplemental materials on the website of the journal in which they were published. Of those 28 studies with replication packages, 11 were not available anymore, probably due to their age. The remaining replication packages were ranked with the following criteria, where the more important criteria has a lower number:

1. Data contains raw eye tracking data.
2. Lowest amount of data loss.
3. Greatest amount of data points.
4. Data quality is superficially good, which means not a lot of data loss at once and not a high number of values that make no sense when looking through the data.

5. A software tool was used that is not already used in the comparison of software tools.

By applying the first four criteria, a list of six replication packages was created. The replication package from McChesney et al. [13] was chosen as it was found to fulfill these criterias best. This is due to the fact that it used Tobii Pro Lab to calculate the fixations, which gives an additional data point in the comparison of different software tools. For example, the data set of study 4 [17] was not chosen as the x and y coordinates consisted of values between -1 and 1 with no indication on how to convert them to a format that works with most software tools, which makes it difficult to provide a comparison between software tools.

4.4 DATA EXTRACTION

The pieces of information that were extracted from each paper can be found in Table 4.1.

Table 4.1: Extracted data items

Category	Data Items
Included Papers	Title, DOI, citation (APA), publication year, venue
Software Tools	software tool used, software tool justifications and consequences
Eye Tracker	eye tracker used, eye tracker configuration
Study Design	study task, item type, item language, item scrolling
Participants	number of participants, demographics of participants
Replication available	replication package

To make sure the data was extracted in a consistent way, a data extraction form was created and used to extract the data. A copy of the form is provided in Table A.1.

The data items were defined with an as good as possible description. Of all data items, only the justifications and consequences of using a specific software tool to analyze the eye tracking data had the need to be processed further. We classified the data using thematic analysis [3] and coded the data with an inductive approach, as we did not want to erroneously 'force' a preconceived result. For this data item, we first extracted the relevant passage as a quote. From this quote, we created labels that are describing as accurately as possible the quote. Finally, categories were built from these labels.

4.5 DATA ANALYSIS

For the first two RQs, we used descriptive statistics to analyze the data. We used the number of papers that had specific characteristics to describe the data for RQ1. For RQ2, we used the number of papers that had specific categories and labels of justifications and consequences to describe the data.

For RQ3, we used the open data set from McChesney et al. [13] to compare the results of different software tools.

The data set from McChesney et al. [13] was analyzed with the following software tools:

- Ogama version 5.1[3]
- PyGaze version 0.6.0[4] with PyGazeAnalyzer version 0.1.0[5]
- EMIP, commit from 2022-09-04[6]

To understand if there is a difference between different software tools, the fixations and saccades of the data set were calculated. As the software tools all have different capabilities, only fixations and saccades were considered. Fixations could be calculated by all named software tools. Saccades could only be calculated by PyGaze and Ogama. The data set already contained the fixations and saccades calculated by Tobii Pro Lab.

iTrace was not used as it does not support import of other data. We tried to convert the data to the format that iTrace uses, but we were not able to do so. The same was true for TAUPE. As neither eyeCode nor PandasEye support the calculation of fixations and saccades, they were not considered for the analysis.

To compare those software tools, the different types of algorithms are named and described in the following section.

- Ogama: fixation detection algorithm from LC Technologies, which is a dispersion-type algorithm with window.
- PyGaze: crude algorithm by PyGazeAnalyser, a submodule of PyGaze, not further described.
- EMIP: IDT fixation algorithm.
- Tobii Pro Lab: As McChesney et al. [13] did not provide the algorithm that was used to calculate the fixations/saccades and as Tobii Pro Lab has multiple available options, it is unknown which algorithm was used.

Additionally, the dataset needed to be slightly converted for the different software tools. As only fixations and saccades were calculated, the data set was converted to a format that only contained the timestamp and the x and y coordinates to be able to import the data to Ogama.

For PyGaze and EMIP, the files were read with a custom python script and converted to the format that was needed for the software tools, which consisted also of timestamp and x and y coordinates. Additionally, a slight modification of the code[7] was needed to be able to read the data from PyGaze, as the data was not in the format that was expected by the script. This does not have implications for the results as the data was not changed in any way.

3 http://www.ogama.net/
4 http://www.pygaze.org/
5 https://github.com/esdalmaijer/PyGazeAnalyser
6 https://github.com/nalmadi/EMIP-Toolkit
7 The function remove_missing of the fixation and saccade detection algorithm returned an array where missing values were removed but the index of the array was not resetted and thus not continuous, therefore causing a crash. This was remedied by changing remove_missing such that the indexes of the array were continuous again.

5 EVALUATION

This chapter evaluates the thesis core claims.

5.1 RESULTS

We can now take a look at the results from the thesis for the different research questions.

5.1.1 *RQ1: Which software tools are used to analyze eye tracking data in software engineering studies?*

To understand which software tools are used by researchers during their studies, it is important to know which eye tracker they used. This is due to the fact that some eye tracker may be incompatible with certain software tools or it may also be easier to just use the integrated software tool of the eye tracker than to convert the data properly for another program.

More than half of the studies used a Tobii eye tracker, as can be seen in Figure 5.1. EyeLink and Gazepoint are the manufacturers that come in second and third place, but with a great difference as EyeLink was used eight times and Gazepoint was used seven times.

Overall, 32 out of 97 studies did not name the software tool that was used to analyze the eye tracking data. 4 out of 97 studies did not use a software tool or algorithm to analyze any part of the eye tracking data.

As can be seen in Figure 5.2, the most-used software tool is Tobii Software [1] with 15 uses, followed by iTrace with 11 uses and Ogama with 10 uses. Except for the software tool Taupe with four uses, all other software tools were only used once. Out of the studies where no tool or algorithm could be properly assigned, five studies used a custom tool that the authors themselves created to analyze the eye tracking data. Two studies used no software tool or algorithm and only did simple analytics. Two studies used no software tool or algorithm and did no analytics at all. These four studies have been collapsed to one bar in Figure 5.2.

Some studies also used algorithms to analyze the eye tracking data, see Figure 5.3. The I-VT filter, the velocity-based algorithm and the Needleman-Wunsch algorithm were used twice, whereas the rest was only used once.

Before 2015, there has not been many eye tracking studies in software engineering. Sharafi et al. [20] found 36 papers in total, and we used 31 of those. From those studies, the most used tools are Tobii Software and Taupe with four uses each. All other named tools were not used more than once. Around 52% did not name the software tool or algorithm they used. Discarding custom tools and tools that were not named, there was a variety of five different tools used.

1 The term Tobii Software encompasses every software tool developed by Tobii. As there exist multiple versions which changed name over time, they were collapsed to this term.

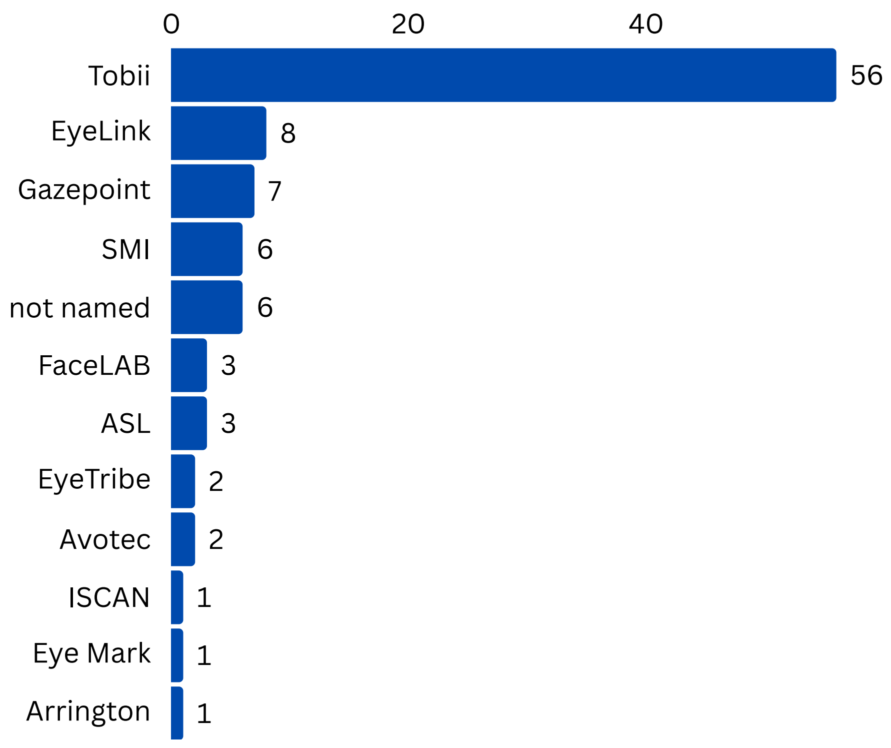

Figure 5.1: Number of brand-specific eye trackers used in software engineering studies from 1990 to 2023

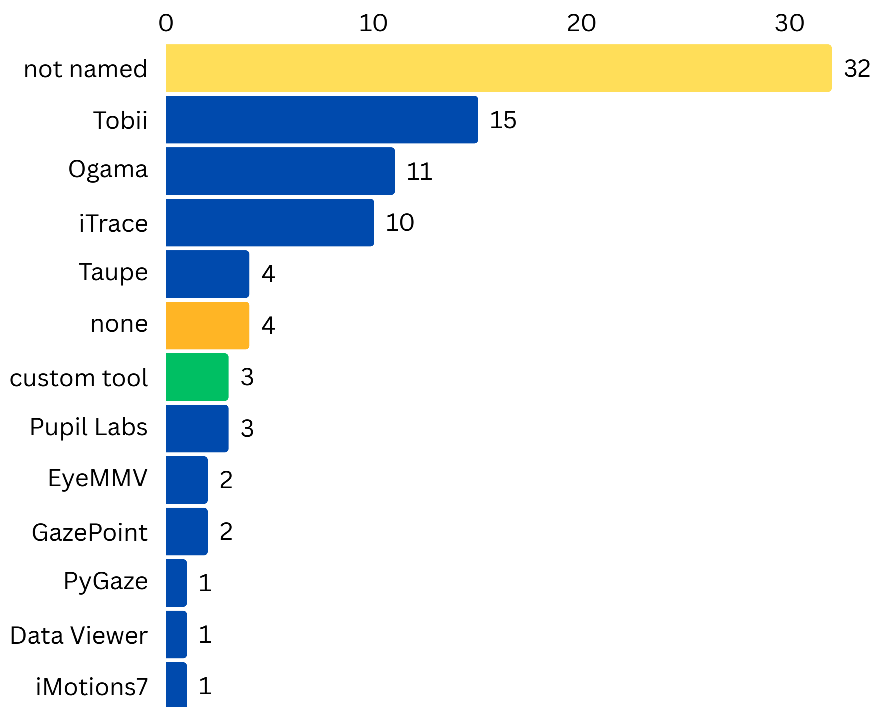

Figure 5.2: Number of type-specific software tools used to analyze eye tracking data in software engineering studies from 1990–2023

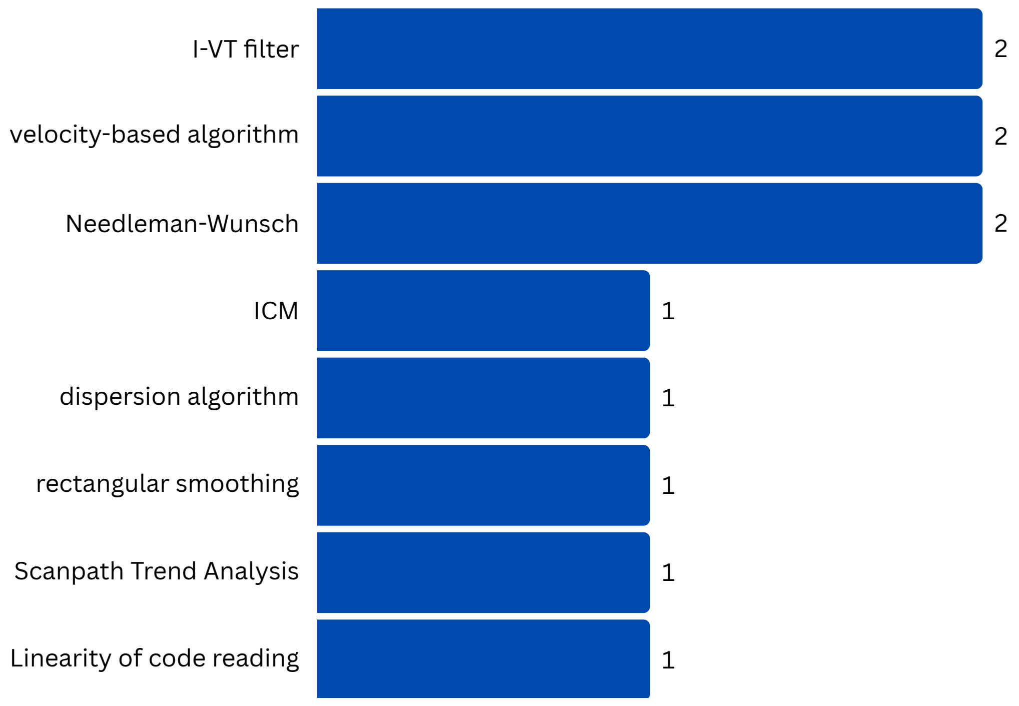

Figure 5.3: Number of different algorithms used to analyze eye tracking data in software engineering studies from 1990–2023

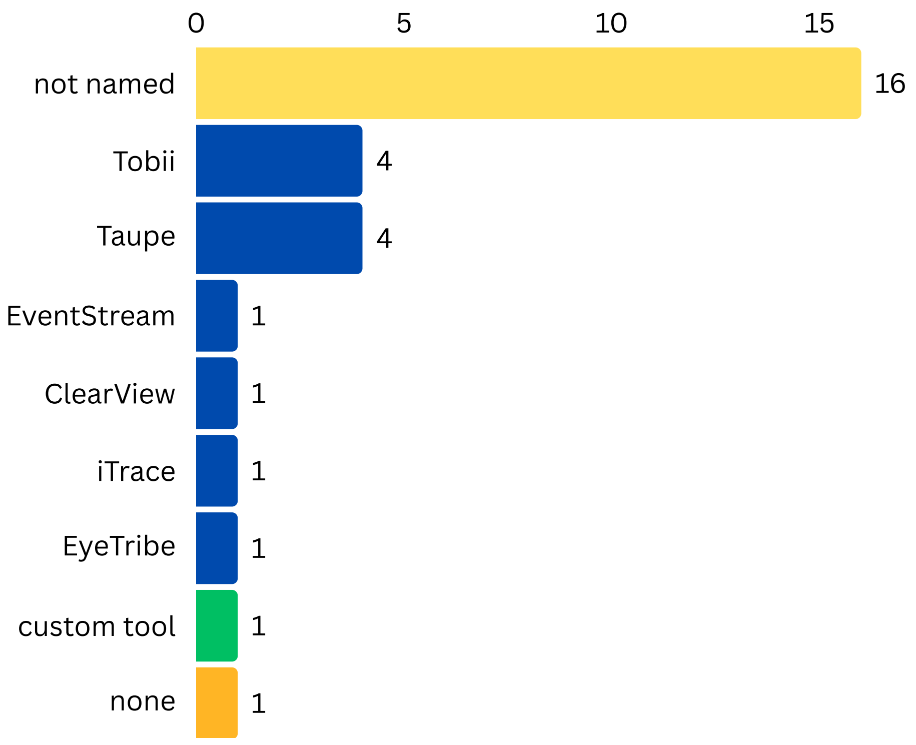

Figure 5.4: Number of type-specific software tools used to analyze eye tracking data in software engineering studies from 1990–2014

Between 2015 and 2019, we found 32 studies in software engineering that used eye tracking, which can be seen in Figure 5.5. Ogama was the most-used software tool to analyze eye tracking data in this timeframe with seven uses, followed by Tobii Software with four uses and iTrace with three. Around 19% did not name the software tool or algorithm they used. Discarding custom tools and tools that were not named, there was a variety of ten different tools used.

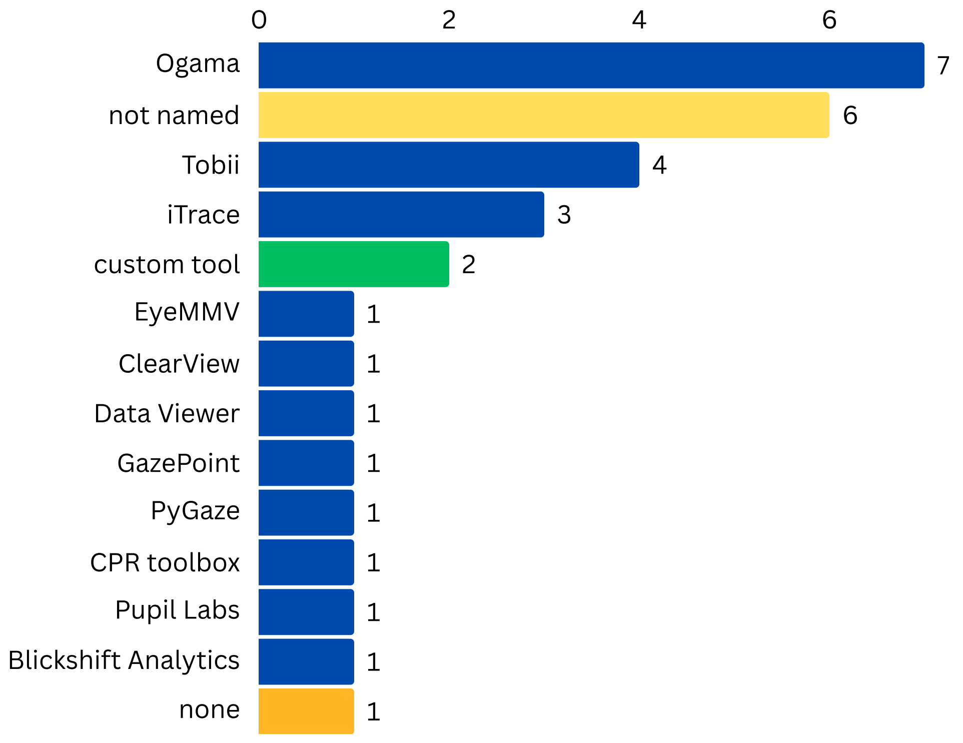

Figure 5.5: Number of type-specific software tools used to analyze eye tracking data in software engineering studies from 2015–2019

Between 2020 and 2023, we found 35 studies in software engineering that used eye tracking, which can be seen in Figure 5.6. Tobii Software and iTrace were the most-used software tools with seven uses each, followed by Ogama with 3 uses. Around 26% did not name the software tool or algorithm they used. Discarding custom tools and tools that were not named, there was a variety of six different tools used.

As different software tools have different features, they can also not calculate each value that may be needed. Therefore, it is interesting to see which values are most needed during eye tracking studies in software engineering. The value that was calculated the most often is the fixation with 44 times, as can be seen in Figure 5.7. It is followed by saccades and AOIs, with both 19 times each. The fixation time and mapping the eye tracking data were also calculated more often with eleven and eight times respectively.

Tobii eye tracker have been used by most studies (around 57%). In Figure 5.8, we can see which software tools were used to analyze the eye tracking data for studies that used Tobii eye tracker. The studies mostly used Tobii Software with fifteen uses, followed by Ogama with ten uses and Taupe with four uses. Around 41% did not name the software tool or algorithm they used in combination with the Tobii eye tracker.

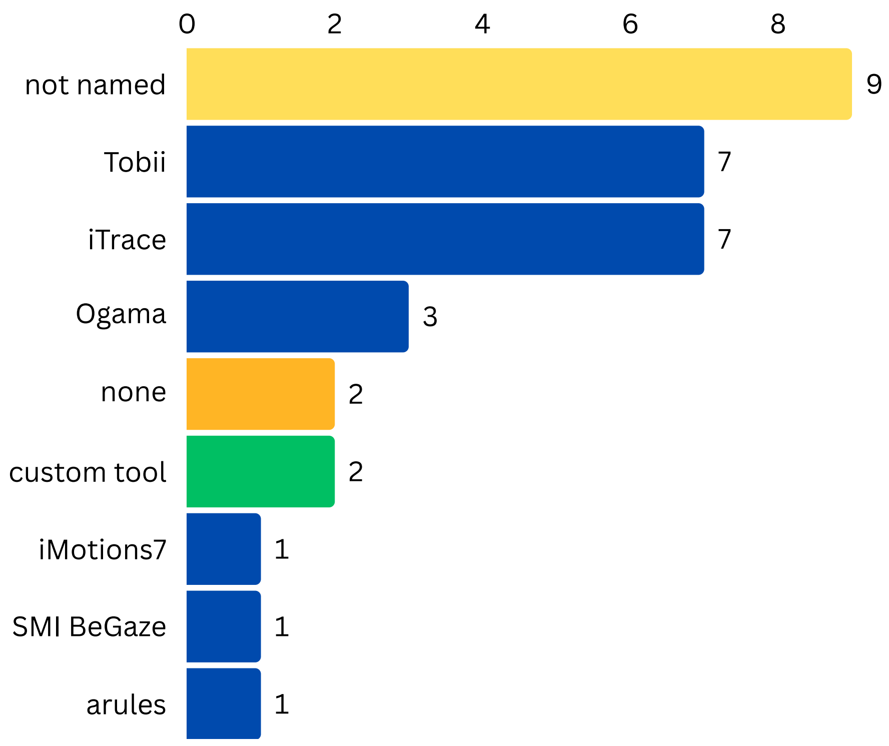

Figure 5.6: Number of type-specific software tools used to analyze eye tracking data in software engineering studies from 2020–2023

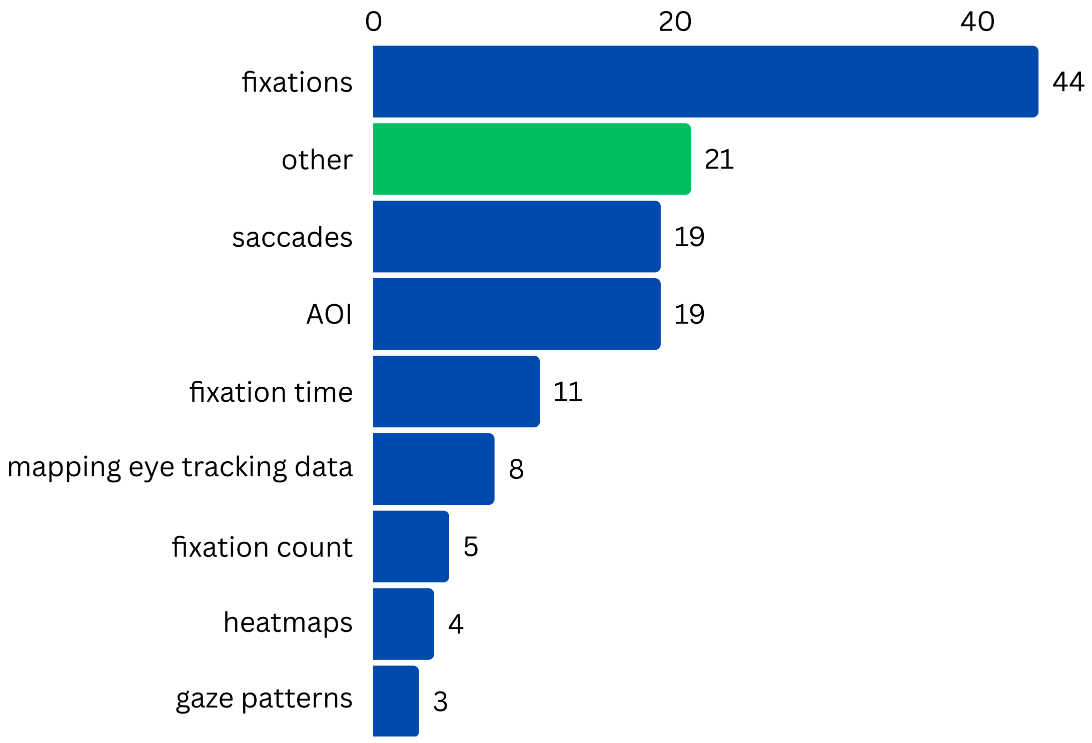

Figure 5.7: Number of different metrics calculated by software tools

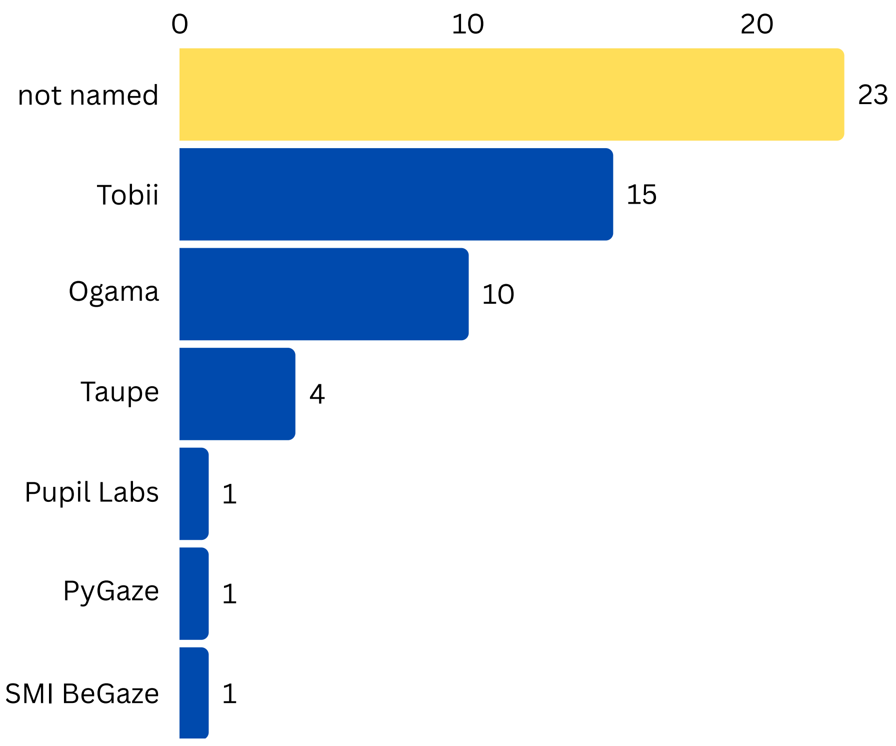

Figure 5.8: Number of type-specific software Tools used by Tobii eye tracker

Fifteen studies allowed the participants to scroll during the eye tracking experiment. The different tools that were used can be seen in Figure 5.9 The most-used tool is iTrace with nine uses, followed by Tobii Software with two uses.

The most-used software tools, Ogama, Tobii Software and iTrace, were categorized depending on which study task was performed. The results can be seen in Figure 5.10 Tobii Software and Ogama were only used for program comprehension and debugging tasks, whereas iTrace has more variety as it also covered non-code comprehension and traceability tasks.

The most important tools and their respective features can be seen in Figure 5.1. Most of the tools are not supported anymore. Only iTrace and Tobii Software are able to support scrolling during the recording, but only in specific windows. Tobii Software is only able to support scrolling in browsers, whereas iTrace has multiple addons to support scrolling in browsers and different Integrated Development Environments (IDEs).

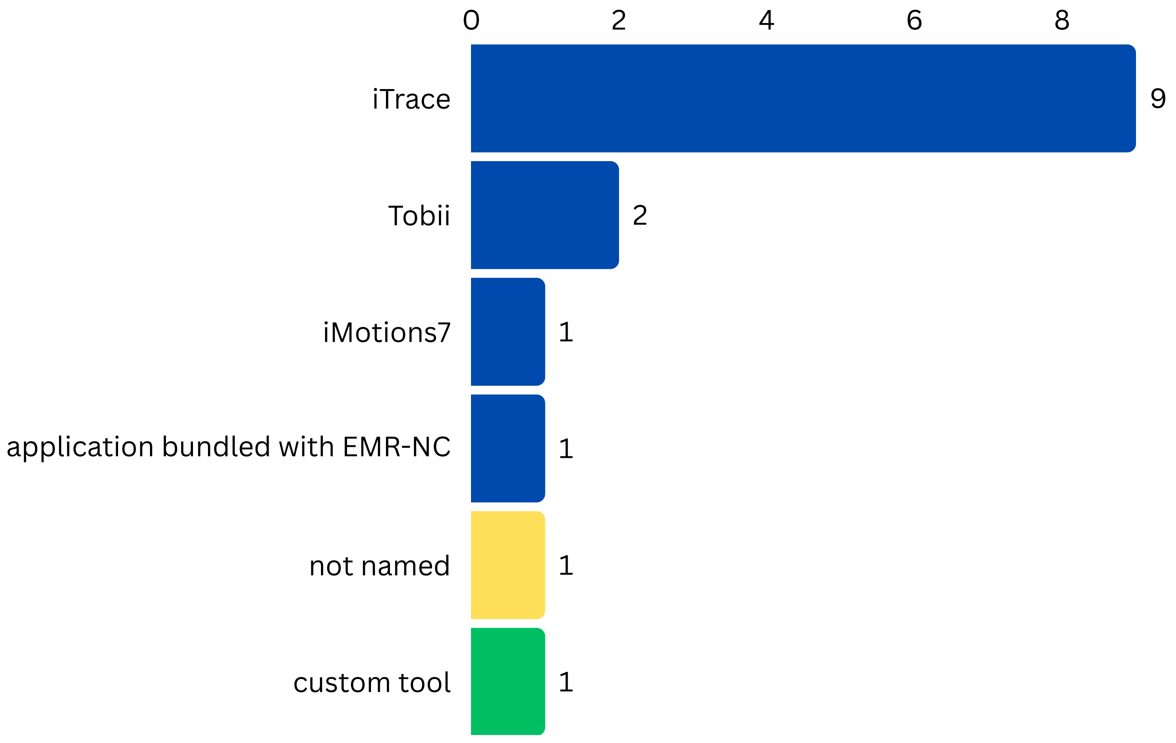

Figure 5.9: Number of type-specific software tools used for items where scrolling is allowed

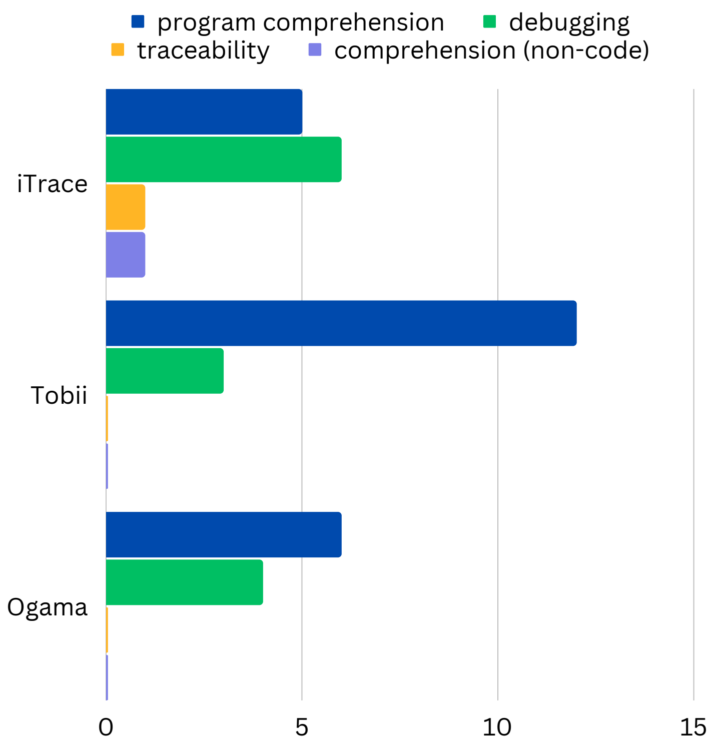

Figure 5.10: Number of different study tasks of iTrace, Tobii and Ogama

Table 5.1: Software tools and their features

Capabilities	Software Tools								
	Ogama	Taupe	iTrace	EyeCode	PandasEye	PyGaze	EMIP	Tobii Pro Lab	iMotions
AOI analysis	yes	yes	yes	yes	no	yes	yes	yes	yes
Plots	yes	yes	yes	yes	no	yes	yes	yes	yes
Metrics	yes	yes	no	yes	no	yes	yes	yes	yes
ML analysis		no	no	no	no	no	no	no	no
Realtime recording	yes	no	yes	no	no	no	no	yes	yes
Support scrolling	no	no	yes	no	no	no	no	partially (browsers)	no
Programming required	no	no	no	yes	yes	yes	yes	no	no
ongoing support	no	no	yes	no	no	yes	no	yes	yes
Hardware compability	yes	partially	partially	yes	yes	yes	no	no	yes
Multi-input integration	yes	no	no	no	no	yes	no	no	yes
open source	yes	yes	yes	yes	yes	yes	yes	no	no

5.1.2 *RQ2: How do authors justify their choices of software tools and what consequences do they discuss?*

During the data collection process, the justifications and consequences the authors named for using a specific software tool were gathered. Those were then mapped to labels, which were then compiled into categories. Out of 97 collected studies, 66 studies did not name any justifications or consequences. The remaining 31 studies were analyzed and the results can be seen in Table 5.2 and Table 5.3.

Table 5.2: Justifications categories

Categories	# of Mentions	# of Labels	Description
visualization	5	4	Mentions of visualization techniques
features	16	8	Mentions of specific abilities of the software tool
non-functional characteristics	8	8	Mentions of characteristics of the software tool that are not of functional nature
support	4	3	Mentions of different ways the software tool supports other hardware/software

In total, four different categories for justifications were identified, which can be seen in Table 5.2. Of those categories, the feature category was mentioned the most and had also the highest amount of labels. The most named label of the feature category was the ability to support scrolling during the eye tracking measurements with five mentions. The second most named label was the ability to map the eye gaze to meaningful elements on the screen with four mentions. The visualization category mostly consisted of the advantage of having different results from the eye tracking measurements visualized such as fixations and more. The non-functional characteristics relate to some characteristics of the software tool such as simplicity of use, ease of modifying the software tool for its own needs, that the software tool is open source and some more. All labels that were identified were unique. The most named label of the support category is the support of several eye trackers. The other labels were support for a macintosh computer and synchronization of the eye tracking data and the data of a Galvanic Skin Response (GSV) sensor.

Table 5.3: Consequences categories

Categories	# of Mentions	# of Labels	Description
features	7	6	Mentions of visualization techniques
non-functional characteristics	4	3	Mentions of specific abilities of the software tool
support	1	1	Mentions of characteristics of the software tool that are not of functional nature

For the consequences, three different categories were identified, which can be seen in Table 5.3. As with the justifications categories, the feature category was the most mentioned one and had the highest amount of labels. Only one label was mentioned twice, which is the inability to collect eye tracking data from outside the IDE window. The other labels were all naming missing features that would have been beneficial for their analysis of the eye tracking data. The non-functional characteristics category had one label, the inaccuracy of the analyzed data, mentioned twice. The other two labels were the difficulty to find optimal parameters to analyze the eye tracking data correctly and that the software tool is closed source. The support category has only one label, which is that there is no synchronization of the measured data between fMRI and an eye tracker.

5.1.3 *RQ3: Which effects on the results and conclusions does using different software tools on the same experiment data have?*

To understand the effects on the results and conclusions of using different software tools on the same experiment data, we used the replication package from McChesney et al. [13]. The replication package contains the eye tracking data from 30 participants that were recorded during a program comprehension task. By using the same data, we can compare the results when analyzed with different software tools. The fixation counts were calculated for the different software tools. The results can be seen in Table 5.4. The Tables 5.5–5.9 will show the average similarity to Tobii Pro Lab and the average variance to provide a summary of the results. The entire tables can be found in the Appendix A.3.

As it can be seen in Table 5.5, no software tool came to the same amount of fixations. PyGaze, Ogama and Tobii Software had a similar amount of fixations, whereas EMIP had a significantly lower amount of fixations. As each software tool used different algorithms to calculate the fixations, except for Tobii Pro Lab, where the algorithm is not known, a difference in the amount of fixations is expected.

PyGaze has the highest similarity to the values found by Tobii Pro Studio with 96.85% whereas EMIP has the lowest similarity with 30,91%. Ogama has a similarity of 90.39% to the values found by Tobii Pro Studio, which is still near to the original values, but it also

Table 5.4: Calculated fixation counts by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	EMIP	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	EMIP / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	2908	612	2841	2970	97.91%	20.61%	95.66%
P114	740	110	819	934	79.23%	11.78%	87.69%
P131	957	113	1208	1331	71.90%	8.49%	90.76%
P157	2101	744	1819	1899	110.64%	39.18%	95.79%
P214	2641	1357	2010	2178	121.26%	62.30%	92.29%
P237	2337	305	2507	2595	90.06%	11.75%	96.61%
P267	2039	768	1738	1895	107.60%	40.53%	91.72%
P270	3031	1223	2729	2973	101.95%	41.14%	91.79%
P316	272	15	469	629	43.24%	2.38%	74.56%
P323	757	140	828	905	83.65%	15.47%	91.49%
P365	2324	952	1586	1731	134.26%	55.00%	91.62%
P370	1617	723	1234	1274	126.92%	56.75%	96.86%
P402	2829	1269	2348	2480	114.07%	51.17%	94.68%
P450	1038	180	1136	1203	86.28%	14.96%	94.43%
P459	1174	167	1320	1477	79.49%	11.31%	89.37%
P469	1391	200	1385	1662	83.69%	12.03%	83.33%
P513	1401	477	1268	1254	111.72%	38.04%	101.12%
P536	1118	237	1030	1291	86.60%	18.36%	79.78%
P561	1178	114	1684	1763	66.82%	6.47%	95.52%
P611	1649	726	1171	1287	128.13%	56.41%	90.99%
P642	1171	162	1027	1257	93.16%	12.89%	81.70%
P645	1325	321	1440	1778	74.52%	18.05%	80.99%
P653	632	47	952	1199	52.71%	3.92%	79.40%
P708	1377	687	1102	1185	116.20%	57.97%	93.00%
P751	3411	1586	2745	2881	118.40%	55.05%	95.28%
P758	988	169	1089	1403	70.42%	12.05%	77.62%
P812	2057	447	1688	1887	109.01%	23.69%	89.45%
P819	2443	1851	2159	2160	113.10%	85.69%	99.95%
P842	2052	376	1833	1892	108.46%	19.87%	96.88%
P900	2567	1323	1891	2069	124.07%	63.94%	91.40%
Average Similarity to Tobii Pro Lab					**96.85%**	**30.91%**	**90.39%**
Average Variance					**19.61%**	**20.09%**	**5.33%**

has the lowest variance with 5.33% to the values found by Tobii Pro Studio, compared to 19.61% for PyGaze and 20.09% for EMIP.

Table 5.5: Total fixation duration similarity by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

	PyGaze / Tobii Pro Lab	EMIP / Tobii Pro Lab	Ogama / Tobii Pro Lab
Average Similarity to Tobii Pro Lab	58.39%	12.31%	124.94%
Average Variance	14.15%	7.60%	10.64%

To understand the difference between the software tools in more detail, we also calculated the total fixation duration and the average fixation duration. The results can be seen in Table 5.5 and Table 5.6.

For PyGaze and EMIP, the total fixation duration is significantly lower than for the other software tools. Especially, EMIP has a total fixation duration that is only 12.31% of the values of Tobii Pro Lab. Ogama's total fixation duration is slightly greater than the one from Tobii Pro Lab with 124.94% and is closest to the values of Tobii Pro Lab.

The average fixation duration is the highest for Ogama with nearly 450% of the values of Tobii Pro Lab. PyGaze and EMIP are, compared to Ogama, relatively close to the values of Tobii Pro Lab with 175% and 126% respectively. The variance of the average fixation duration of all participants is significantly higher for Ogama with nearly 200%, whereas PyGaze and EMIP have a variance of around 50%.

Table 5.6: Average fixation duration similarity by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

	PyGaze / Tobii Pro Lab	EMIP / Tobii Pro Lab	Ogama / Tobii Pro Lab
Average Similarity to Tobii Pro Lab	178.33%	129.15%	445.18%
Average Variance	50.48%	48.75%	196.94%

Only PyGaze and Ogama were able to calculate the saccades, as can be seen in Table 5.7. The number of saccades is greatly different between all three different software tools. The difference is especially high between PyGaze/Ogama and Tobii Pro Lab. This is due to the fact that Tobii Pro Lab counts multiple saccades between different fixations, therefore the number of saccades is higher. Disregarding this fact, the difference between the software tools is probably due to the different algorithms that were used and the differences in the parameters that were used for the algorithms. As McChesney et al. [13] did not state which parameters were used, it is not possible to compare them to those of PyGaze and Ogama. Additionally, Ogama uses needs some more parameters than PyGaze, which could also lead to a difference in the results. Nevertheless, it can be seen that a difference in parameters and algorithms can lead to a difference in the results.

Like we did for the fixations, we also calculated the total saccade duration and the average saccade duration. The results can be seen in Table 5.8 and Table 5.9. The total saccade duration is significantly higher for PyGaze and Ogama than for Tobii Pro Lab with 1253% and 497% respectively. This is due to the fact that Tobii Pro Lab counts multiple saccades

Table 5.7: Calculated saccades counts similarity by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

	PyGaze / Tobii Pro Lab	**Ogama / Tobii Pro Lab**
Average Similarity to Tobii Pro Lab	9.56%	34.27%
Average Variance	2.93%	10.10%

between different fixations, therefore the number of saccades is higher and the total saccade duration is lower. There is still a rather big difference in total saccade duration between PyGaze and Ogama, which can be attributed to the different amount of saccades they found.

Table 5.8: Total saccade duration similarity by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

	PyGaze / Tobii Pro Lab	**Ogama / Tobii Pro Lab**
Average Similarity to Tobii Pro Lab	1253.80%	497.23%
Average Variance	248.86%	231.62%

For the average saccade duration metric, PyGaze stands out by having a significantly higher average saccade duration than the other software tools with 5003.21% of the values of Tobii Pro Lab. It has also around 10 times the average saccade duration of Ogama. This may be due to the fact that the parameters could not be correctly adjusted in PyGaze, as the values of the required arguments could not be found in the documentation of the eye tracker. Ogama also has a significantly higher average saccade duration than Tobii Pro Lab with 426.00% of the values of Tobii Pro Lab.

Table 5.9: Average saccade duration similarity by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

	PyGaze / Tobii Pro Lab	**Ogama / Tobii Pro Lab**
Average Similarity to Tobii Pro Lab	5003.21%	570.91%
Average Variance	1038.74%	294.35%

5.2 DISCUSSION

In this section, we will discuss the previously presented results. We will go through all three research questions and discuss the results for each of them.

5.2.1 *Software Tool Usage*

One motivation of this thesis was to find out which software tools are used to analyze eye tracking data in software engineering studies. Especially, we wanted to find out if most studies named the software tool they used and how the usage of different software tools changed over time.

We found that around 32% of the studies did not name the software tool they used to analyze the eye tracking data. This is a relatively high percentage, as it is important to know which software tool was used to analyze the eye tracking data, such that the results can be reproduced. It is important to report the software tool used, as different software tools may have different algorithms to analyze the eye tracking data and thus may come to different results, as can be seen in the results of RQ3. It is also important to know which software tool was used, as some software tools may not be able to analyze the eye tracking data in the way the researcher wants to. For example, if a researcher wants to analyze the eye tracking data in real-time, but the software tool does not support this, the researcher would have to find another software tool that supports this or would have to change the research question. Another example would be if the software tool does not support the eye tracker that the researcher wants to use. Before 2014, around 52% of the studies did not name the software tool they used. The percentages were lowered in the following years, as between 2015 and 2019, around 19% of the studies did not name the software tool they used and between 2020 and 2023, around 26% of the studies did not name the software tool they used. This indicates that the reporting of the software tool used to analyze the eye tracking data has improved over time, even though it has dropped a bit in the last few years. This is probably due to the increased interest in eye tracking studies in software engineering and the resulting need for a specific reporting structure that required the software tool used. Holmqvist et al. [8] presented in their paper a reporting structure for eye tracking studies. Among the strongly recommended aspects to be reported were the adequate description of the data processing and analysis steps with all relevant parameters and the reporting of firmware and software versions where applicable..

Another important result is that the most-used software tool is Tobii Software. This is probably due to the fact that Tobii is the most-used eye tracker in software engineering studies. The ease of using the provided software tool of the eye tracker instead of searching a compatible software tool and maybe needing to make some modifications to the data set may also be a reason for this. Another reason may be that the researchers are not aware of other software tools that may be better suited for their needs. This may imply that there may be better software tools available than Tobii Software, but most researchers do not research properly if such a software tool exists.

iTrace being the second most-used software tool is probably due to the fact that it supports scrolling during the eye tracking measurements, which is a feature that is not supported by many other software tools. This makes it a desirable software tool for software engineering studies, as scrolling is a common task in software engineering and increases the validity of a study, as the participants can perform the task in a more natural way. It makes also the design of experiments easier as the participant is not confined onto one screen and can freely scroll and change windows, enabling wider possibilities of experiment setups. iTrace being the only tool to support scrolling besides Tobii Software, which only supports scrolling in specific windows, makes it a valid choice for many studies. Also the continuous support and development of new features, such as the support for VSCode, may be a reason for its popularity.

Ogama being the third most-used software tool is probably due to the fact that it is open source, supports many different eye trackers and has been developed in 2007, which makes it one of the older software tools. Therefore, it is established in the eye tracking community and is probably known by many researchers. Additionally, it was published before the amount of eye tracking studies started to increase due to lower price and greater affordability of eye trackers, which made it a good choice as a proven software tool. But the trend shows that Ogama is used less and less, as between 2015 and 2019, it was the most used software tool, but between 2020 and 2023, it was only the third most used software tool. This is probably due to the fact that it is no longer supported since May 2016. Newer eye trackers may not be fully supported, additionally the algorithms to calculate the eye tracking metrics will also age quickly.

The analysis of software tools used over time showed that there was a lot of variety in the software tools used. Until 2014, mostly Taupe was used, which is probably due to the fact that it was the only open source software tool available at that time. iTrace was just published in 2014, which is probably why it was only used once. Taupe was not used in further studies. This is likely a result of the fact that it only supports the Gazetracker and EyeLink II eye trackers, which are not used in many studies. It is also not supported anymore and therefore not a good choice for new studies.

Between 2015 and 2019, Ogama was the most used software tool, followed by Tobii Software and iTrace. In this timeframe there was a lot of variety of software tools used with ten different software tools used. During this time, the amount of eye tracking studies in software engineering increased, which may be a reason for the variety of software tools used. There was a need for new features and also new eye trackers were used, which increased the need for new software tools.

Between 2020 and 2023, iTrace and Tobii Software were the most used software tools, followed by Ogama. In this timeframe, there was less variety of software tools used with only six different software tools used. This can be explained by the fact to the fact that an equilibrium was found between the software tools that are used. The software tools that are used are well established and have been used in many studies, which makes them a good choice for new studies. iTrace being the second most used software tool is probably due to the fact that it supports scrolling during the eye tracking measurements, which is a feature that is not supported by many other software tools and is a desirable feature for software engineering studies. As Tobii was also the most used eye tracker manufacturer in this timeframe, it is not surprising that Tobii Software was the most used software tool.

Twelve studies did not use a software tool but an existing algorithm to analyze the eye tracking data. Some of them are algorithms that are also used by some software tools, so it may be that they just named the algorithm that was used and not the software tool that used it. This may have implications for the results as there could be implementation-specific differences in the algorithm which could lead to different results, depending on the software tool. Therefore such studies should rather name the software tool, if they used one, or add the code for their implementation such that it can be checked for differences.

It is not surprising that only four studies did no or only simple analytics, as the reason for choosing to do an eye tracking experiment is to use and analyze the eye tracking data. As the eye tracking data can also be used to replay how a person looked on the screen, it does not need to be always analyzed. This shows us that there are more ways eye tracking data can be used than just objective analysis.

Five studies used custom tools to analyze eye tracking data. This is probably mostly due to the vast amount of work to create a custom tool, which may only be used once for this study. Additionally, it reduces the reproducibility of the study as no one else can use this custom tool. Therefore it is mostly not a good choice, only when the study has some very specific needs that can not be met with other software tools.

The values that were most calculated by different software tools were fixations, saccades and AOIs. As not every study named the values they calculated, there may be other values that were calculated more often. Nevertheless, it seems as if fixations are the most important value of eye tracking data. This is probably due to the fact that fixations are the most basic value of eye tracking data and are needed to calculate other values, such as saccades. Additionally, fixations are a good indicator of where the participant was looking at, which is important for many studies. Interestingly there were not as many fixations named as saccades, even though these are mostly calculated together. This may be because not all studies named all metrics that they calculated. There is also a wider variety of calculated values by software tools, most of which were only calculated once. This may be attributed to the fact that different studies have different needs and therefore need different values to be calculated.

By looking at the software tools used by Tobii eye tracker, we can see that Tobii Software and Ogama were the most used software tools. It is not surprising that Tobii Software was the most used software tool, as it is the software tool provided by Tobii. Interestingly, around 41% of the studies did not name the software tool they used in combination with the Tobii eye tracker. Probably most of those studies did use the Tobii Software software tool, but did not name it, as they probably thought it was not necessary to name it, as it is the software tool provided by Tobii. Even though it may make sense to them, it is not clear for other persons reading the papers and therefore should also be named in any case.

As we saw in Figure 5.10, the most-used software tools, Ogama, Tobii Software and iTrace, were categorized depending on which study task was performed. For both Ogama and Tobii Software, the study tasks were program comprehension and debugging. Only iTrace also covered non-code comprehension and traceability. This is probably due to missing features in Ogama and Tobii Software, which are needed for specific studies. For example, traceability is a study task that is not supported by Ogama and only partially by Tobii Software, which is probably why iTrace was used for those studies.

5.2.2 *Justifications and Consequences*

Most studies did not name any justifications or consequences for their choice of software tool. This is probably due to the fact that they did not think it would be important to name them for their choice of software tool, as they probably thought that the software tool they used would not make a difference in the results. It could also be that they did not know any other software tools and therefore did not have a choice, which they did not think to write down. Another reason could be that they had some justifications or consequences to chose a certain software tool, but did not think about writing down them for their choice. Still, it is important to name the justifications or consequences for the choice of software tool, as different software tools may have different algorithms to analyze the eye tracking data and thus may come to different results, as can be seen in the results of RQ3.

Interestingly, there were in total 33 total mentions of justifications and only 12 total mentions of consequences. Researchers probably mostly spoke about why a specific software tool was chosen, but did not think about the consequences of their choice. It may also be that there are not many consequences for the choice of software tool, as most software tools have similar features and therefore do not have many consequences for the choice of software tool.

Justifications

The most mentioned category of justifications for the choice of software tool was the features of the software tool with scrolling support being the most mentioned feature. This result shows that scrolling support is an important feature for software engineering studies. This is probably due to the fact that scrolling is a common task in software engineering and increases the validity of a study, as the participants can perform the task in a more natural way. Software engineers often scroll through code to find a specific part of the code or to get an overview of the code. Additionally, no programming today is done on a static screen, which makes scrolling an important task in software engineering. Therefore, it is not surprising that scrolling support is the most named feature for the choice of software tool.

The second most mentioned category of justifications for the choice of software tool was the ability to map the eye gaze data to meaningful elements. This makes analysis easier and more meaningful, as the eye gaze data can be mapped to the elements that the participant was looking at. Accordingly, the ability to map the eye gaze data to meaningful elements is an important feature for software engineering studies and it is no wonder that it was the second most mentioned feature for the choice of software tool.

Non-functional characteristics were the second most mentioned category of justifications for the choice of software tool. Interestingly, all mentioned labels were unique, which shows that most studies wanted specific characteristics for their choice of software tool. This is probably due to the fact that different studies have different needs and therefore need different characteristics for their choice of software tool. But that also insinuates that there is no software tool that fulfills all the needs of the studies, which may be a reason for the variety of software tools used. Additionally, as all labels are unique, it is not possible to generalize them and therefore it is not possible to say which characteristics are important for software engineering studies. Interestingly, there was only one mention of using a tool because it is open source. This may be due to the fact that there are several open source tools available and therefore it is not a unique characteristic anymore.

Visualizations were the third most mentioned category of justifications for the choice of software tool. Good visualizations abilities are important for eye tracking studies, as they make it easier to to interpret the data and understand the results. This can be seen by the fact that the only label mentioned more than once was the visualization of different results of eye tracking data.

The last category of the justifications, support, shows that hardware and software support are important for the choice of software tool. The only label that was mentioned more than once was the support of several eye trackers. This is probably due to the fact that different eye trackers are used in software engineering studies and therefore it is important that the software tool supports the eye tracker that is used in the study. If only a specific brand of eye trackers is supported, it will limit the choice of eye trackers for the study, which may be undesirable and may compromise the validity of the study. Therefore, it is important that the software tool supports several eye trackers.

Consequences

As with the justifications, the most frequently cited category of consequences for the choice of software tool was the features of the software tool. Only one label was mentioned more than once, which was the inability to collect eye tracking data from outside the IDE window. This shows us that studies want to increase the validity of their studies by running the experiment in a more natural environment, but the software tool is a limitation of this goal. As the other labels were unique, it shows that different studies have different needs and therefore need different features for their choice of software tool. Additionally, it means that the most important features of software tools have been met, as otherwise a label would have been mentioned more than once.

The second most mentioned category of consequences for the choice of software tool is the non-functional characteristics. Only the label "inaccuray of the analyzed data" was mentioned more than once, which shows that the accuracy of the analyzed data is an important characteristic for software engineering studies. Such inaccuracies can lead to wrong results and therefore wrong conclusions, which may compromise the validity of the study. Therefore, it is important that such a consequence is discussed for the choice of a software tool. Interestingly, there was only one mention of a consequence being that a software tool is proprietary. As a lot of people used proprietary software tools such as Tobii Software, a larger amount of this consequence would have been expected. It seems like researchers did not think about the implications of using proprietary software tools and thus did not name it as a consequence of using proprietary software.

The last category of the consequences, support, only has one label in total, which is the lack of synchronization between the eye tracker and fMRI. In combination with the justification of using a specific software tool because it supports synchronization with a GSV sensor, it shows that synchronization is an important feature for software engineering studies. Especially in the current time, where multimodal studies are becoming more and more popular, it is important that the software tool supports synchronization with other sensors such as fMRI, Electroencephalogram (EEG), GSV, . . .

5.2.3 *Software Tool Effects*

The last research question was about the effects of different software tools on the results of eye tracking data. We wanted to find out if different software tools lead to different results and if so, how big the difference is.

The fact that we were able to only compare three software tools shows that there is a great difference in the abilities of software tools that analyze eye tracking data. Not all of them were able to analyze fixations and saccades, which are the most basic values of eye tracking data. Some of them were not able to import data from other software tools or only with a lot of effort by reverse engineering the data format. Especially iTrace is only able to work with data that it has recorded itself, which is unfortunate, as it is the second most used software tool and one of the only software tools that has still ongoing development and support. This makes comparability between different software tools and thus different studies difficult, as the data needs to be converted to the format of the software tool that is used for the analysis.

Of the three open-source software tools that we were able to compare, none had the same algorithms to analyze the eye tracking data. This explains at least some difference between the results. The number of fixations was pretty on par between Tobii Pro Lab, PyGaze and Ogama, contrary to those of EMIP. This may be either due to an error on our side due to a data conversion error or due to the fact that EMIP uses a different algorithm to calculate the fixations, which may also be flawed.

If we take a look at the total fixation duration and the average fixation duration, we can see that there is a big difference between the software tools, also in the case of PyGaze and Ogama. Therefore, it seems as if the algorithms and/or the implementation of the algorithms are different between the software tools. The algorithms that were seen as the most accurate have also changed over time, which is probably why there is a lot of different algorithms used by software tools, especially the older ones, as most of them have no more ongoing development and support. This may also be a reason for the difference in the results, as the algorithms may be outdated and therefore not as accurate as newer algorithms.

For the number of saccades, the differences are even greater than for the number of fixations. For Tobii Pro Lab, this is in part due to the fact that they consider multiple saccades between different fixations, which is not done by PyGaze and Ogama. Therefore, the number of saccades is higher, which explains the difference in the results. Nevertheless, there is still a rather big difference in the number of saccades between PyGaze and Ogama. The reasons for this may be the same as for the number of fixations, as the algorithms and/or the implementation of the algorithms are different between the software tools.

The total saccade duration and the average saccade duration are also significantly different between the software tools, with value differences between Tobii Pro Lab and PyGaze reaching up to 5000%. The variance of those results is also very high, with values reaching up to 1000%. It seems like the software tools calculate the saccade duration in a very different way, which leads to the high differences in the results.

All in all, this leads to the conclusion that different software tools may lead to different results. Especially when different algorithms are used, the results may be very different. This is an issue because it makes it difficult to compare the results of different studies, as the results may be different due to the different software tools used. Even though we only analyzed basic values of eye tracking data, the results were already very different. It is unknown if the results would be even more different for more complex values of eye tracking data, but we can assume that it does make a difference.

This is important knowledge for researchers, as they need to be aware of the fact that different software tools may lead to different results and therefore need to be careful when comparing the results of different studies, especially when different software tools were used. It is also important for researchers to name the software tool they used to analyze the eye tracking data, as it may impact the results and therefore the conclusions of the study.

It also clearly shows the need for a standardized procedure to conduct eye tracking studies. Having too many different software tools available will make it more difficult to compare the results of different studies, as some studies may use different software tools. Therefore, it would be important to have a standardized software tool that is used by most studies, such that the results can be compared more easily. Another possibility would be to have a standardized data format for eye tracking data, such that the data can be easily converted to the format of the software tool that is used for the analysis. This would make it easier to compare the results of different studies, as the data would be in the same format and therefore the results would be more comparable. The trend is already going towards less available software tools, as the number of software tools used in software engineering studies decreased over time. But there is still a lot of variety in the software tools used. Older software tools are still used, as newer might not have all features that are necessary for a study. Implementing these abilities in newer software tools may help to reduce the number of software tools used.

6 THREATS TO VALIDITY

There are several threats that can affect the validity of this thesis.

In the search and selection approach, there may have been papers that were missed due to insufficient search parameters. To minimize this threat, the chosen search parameters are close to the search parameters of Sharafi et al. [20].

Additionally, the restriction of the search to the libraries of IEEEXplore and ACM poses a threat to validity. Some papers that would have been found in other libraries or indexers were probably excluded. This is, unfortunately, due to the smaller scope that this thesis can encompass without being too much work.

The fact that no snowballing was done after the keyword search, compared to Sharafi et al. [20], poses also another threat, which is reduced by finding in total 97 papers, which is a large enough size such that missed papers will not threaten the validity of the study as much as if only a small amount of papers would have been found.

The fact that the search is restricted to the years 2015–2023 is also a threat to validity. The rationale is that the papers that would be interesting for this thesis would have been found by Sharafi et al. [20], as the results of the literature search here are a subset of those from Sharafi et al. [20].

The only criteria that aids in maintaining the quality of the studies is I1. No further quality assessment was performed, even if it is sometimes suggested in guidelines for systematic literature reviews [10]. This thesis aims to find out if there are still flaws in the reporting for eye tracking studies, therefore, such flaws are even "desired".

In the data extraction and analysis process, there may be human errors as only one person performed the search process. We extracted the data ourselves, although it is not recommended by the standards of a SMS study [14]. Due to the way a bachelor thesis is structured, we could not ask other persons to help in extracting the data, which would have increased the validity of this thesis.

Additionally, the data extraction form may be poorly designed which would produce wrong or incomplete results. This threat was diminished by revising the form with the help of the advisors. The data extraction, especially subjective categories such as the justifications and consequences of using a specific software tool, is a threat to validity as only one person identified the extracted data. To minimize this threat, in cases where we were not sure, there were frequent talks with the advisors about the data extraction and analysis process to achieve a high standard.

The comparison of the software tools harbors also threats to validity. The difference between the algorithms that were used may affect the results in such a way that the conclusion that was drawn may be different. This threat could not be diminished as we could not choose to use the same algorithm for all tools. As we did not have a greater pool of software tools that we could choose from for our analysis, we needed to use software tools with different algorithms.

Another threat to validity is that the results may not be generalizable to all software tools. The software tools that we were not able to compare may not provide the same results as the ones that we compared. This threat could not be diminished as we could not compare all software tools that were used in the studies.

The fact that we only compared the software tools with the same data may also be a threat to validity. We only used one data set for the comparison of the software tools, which may also have skewed data and thus skewed the results of RQ3. We may have come to a different conclusion if we used a different data set or if we analyzed more data sets. This threat could not be diminished due to the limited time available for this thesis. More data sets would have considerably increased the effort, as most data sets are saved in different data formats, thus the data extraction process would have been more difficult and time-consuming.

7 CONCLUDING REMARKS

In this chapter, we will summarize our work and also provide an outlook for future work in the direction of eye tracking studies in software engineering.

7.1 CONCLUSION

In this thesis, we explored the current state of art of eye tracking software tools used in software engineering and compared different eye tracking software tools with respect to their results.

By performing a modified SMS on studies published between 2015 and 2023 with the addition of previously found papers by Sharaifi et al. [20], we were able to come to some interesting conclusions. First of all, we found that there has been an increase in the naming of the software tools that were used in the studies over time, but there are still too many studies that do not name the software tools that were used. We also found that the most used software tools are Tobii Studio, iTrace, and Ogama by a relatively large margin. Additionally, the number of different software tools that were used in the studies has decreased over time, while also having multiple open-software tools stop development and support. The most important measures that were calculated with software tools were the fixations, saccades and AOIs. We also found that iTrace has been used a lot because it is one of the few software tools to support scrolling during eye tracking measurements.

We also analyzed the justifications and consequences that authors named as to why they chose a certain software tool. We found that scrolling support was an important argument as to why a software tool was used. Additionally, it seems like the most important non-functional characteristics that a software tool should have are already present. Other important justification categories were visualizations and support for software and hardware.

The greatest consequence named was that the software tool was unable to collect eye tracking data from outside the IDE window. Other categories of named consequences were non-functional characteristics and support for software and hardware.

Lastly, we compared fixations and saccades and some of their respective metrics calculated by different software tools. We compared the results of Ogama, PyGaze, EMIP and Tobii Pro Lab. In doing so, we were able to find out that there exists a difference in the results, at times those results were even greatly different from each other. While the software tools had different algorithms that they used to calculate the results, it is still an interesting finding and one that needs to be further researched.

In conclusion, we can see that there is a need for a standardized data analysis process to ensure the comparability of eye tracking studies in software engineering.

7.2 FUTURE WORK

In this section, we will describe which possibilities of future work have emerged from this thesis.

Eye Tracking Studies

The results of RQ1 and RQ2 from this thesis have shown us the current state of art of eye tracking software tools used in eye tracking. It would be interesting to replicate these research questions in around five years to see the development of the usage of software tools in the near future. This is an especially interesting point as the number of eye tracking studies in software engineering have been increasing rapidly the last 10 years. Sharafi et al. [20] found 36 papers from 1990 to 2014 while this thesis found 97 papers from 2015 to 2023. Therefore, we recommend a replication in around five years, as we expect that the number of found papers will be even greater than in this time frame.

Software Tools Comparability

As already described in the Threats to Validity, we were only able to compare software tools with different types of algorithms. A new study that investigates the impact of the software tools and their respective configurations would help in providing a better understanding of the differences in the results between the software tools. Conducting a study with more or even all currently used software tools will also provide better basic knowledge. Additionally, exploring more metrics such as AOIs and their dependent metrics will show the differences in more detail. It will also enable to investigate if the conclusions that studies formed from their eye tracking data are consistent among different software tools or if the choice of software tool may influence the result. The outcome of such a study could be of great importance for the community.

Standardized Data Analysis Procedure

From the results of this thesis, it can be seen that there is a need for a standardized procedure to analyze eye tracking data. Having no consistent data format for the data sets of eye tracking data has a negative impact on the reproducibility of the results of studies. Also, a lot of studies use proprietary software to analyze their eye tracking data, which makes it more difficult to compare the data if one has no access to this software. Therefore, we recommend that a specific data format is chosen to be used by every eye tracker and that the research community should agree on a specific open-source software tool to use to analyze the eye tracking data.

APPENDIX A

A.1 APPENDIX A: ANALYZED STUDIES

[S1] Amine Abbad-Andaloussi, Thierry Sorg, and Barbara Weber. “Estimating Developers’ Cognitive Load at a Fine-Grained Level Using Eye-Tracking Measures.” In: *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. ICPC ’22. Virtual Event: Association for Computing Machinery, 2022, 111–121. ISBN: 9781450392983. DOI: 10.1145/3524610.3527890. URL: https://doi.org/10.1145/3524610.3527890.

[S2] Nahla J. Abid, Bonita Sharif, Natalia Dragan, Hend Alrasheed, and Jonathan I. Maletic. “Developer Reading Behavior While Summarizing Java Methods: Size and Context Matters.” In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 384–395. DOI: 10.1109/ICSE.2019.00052.

[S3] Maike Ahrens, Kurt Schneider, and Melanie Busch. “Attention in Software Maintenance: An Eye Tracking Study.” In: *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*. 2019, pp. 2–9. DOI: 10.1109/EMIP.2019.00009.

[S4] Zubair Ahsan and Unaizah Obaidellah. “Predicting expertise among novice programmers with prior knowledge on programming tasks.” In: *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2020, pp. 1008–1016.

[S5] Zubair Ahsan and Unaizah Obaidellah. “Is Clustering Novice Programmers Possible? Investigating Scanpath Trend Analysis in Programming Tasks.” In: *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*. ETRA ’23. Tubingen, Germany: Association for Computing Machinery, 2023. ISBN: 9798400701504. DOI: 10.1145/3588015.3589193. URL: https://doi.org/10.1145/3588015.3589193.

[S6] Naser Al Madi. “How Readable is Model-Generated Code? Examining Readability and Visual Inspection of GitHub Copilot.” In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE ’22. Rochester, MI, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3560438. URL: https://doi.org/10.1145/3551349.3560438.

[S7] Nasir Ali, Zohreh Sharafl, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. “An empirical study on requirements traceability using eye-tracking.” In: *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE. 2012, pp. 191–200.

[S8] Magdalena Andrzejewska and Anna Stolińska. “Do Structured Flowcharts Outperform Pseudocode? Evidence From Eye Movements.” In: *IEEE Access* 10 (2022), pp. 132965–132975. DOI: 10.1109/ACCESS.2022.3230981.

[S9] Christoph Aschwanden and Martha Crosby. “Code scanning patterns in program comprehension.” In: *Proceedings of the 39th hawaii international conference on system sciences*. Citeseer. 2006.

[S10] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. “Do Developers Read Compiler Error Messages?” In: *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 2017, pp. 575–585. DOI: 10.1109/ICSE.2017.59.

[S11] J Bauer, J Siegmund, N Peitek, JC Hofmeister, and S Apel. “Indentation: simply a matter of style or support for program comprehension? In 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC).” In: *IEEE* 2 (2019), pp. 14–45.

[S12] Roman Bednarik. “Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations.” In: *International Journal of Human-Computer Studies* 70.2 (2012), pp. 143–155.

[S13] Roman Bednarik and Markku Tukiainen. “An eye-tracking methodology for characterizing program comprehension processes.” In: *Proceedings of the 2006 symposium on Eye tracking research & applications*. 2006, pp. 125–132.

[S14] Tanya Beelders. “Eye-Tracking Analysis of Source Code Reading on a Line-by-Line Basis.” In: *Proceedings of the Tenth International Workshop on Eye Movements in Programming*. EMIP ’22. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2022, 1–7. ISBN: 9781450392891. DOI: 10.1145/3524488.3527364. URL: https://doi.org/10.1145/3524488.3527364.

[S15] Andrew Begel and Hana Vrzakova. “Eye Movements in Code Review.” In: *Proceedings of the Workshop on Eye Movements in Programming*. EMIP ’18. Warsaw, Poland: Association for Computing Machinery, 2018. ISBN: 9781450357920. DOI: 10.1145/3216723.3216727. URL: https://doi.org/10.1145/3216723.3216727.

[S16] Ian Bertram, Jack Hong, Yu Huang, Westley Weimer, and Zohreh Sharafi. “Trustworthiness Perceptions in Code Review: An Eye-Tracking Study.” In: *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ESEM ’20. Bari, Italy: Association for Computing Machinery, 2020. ISBN: 9781450375801. DOI: 10.1145/3382494.3422164. URL: https://doi.org/10.1145/3382494.3422164.

[S17] Dave Binkley, Marcia Davis, Dawn Lawrie, Jonathan I Maletic, Christopher Morrell, and Bonita Sharif. “The impact of identifier style on effort and comprehension.” In: *Empirical software engineering* 18 (2013), pp. 219–276.

[S18] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. “Eye Movements in Code Reading: Relaxing the Linear Order.” In: *2015 IEEE 23rd International Conference on Program Comprehension*. 2015, pp. 255–265. DOI: 10.1109/ICPC.2015.36.

[S19] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. “What influences dwell time during source code reading? Analysis of element type and frequency as factors.” In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. 2014, pp. 335–338.

[S20] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. "Analysis of code reading to gain more insight in program comprehension." In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. 2011, pp. 1–9.

[S21] Nergiz Ercil Cagiltay, Gul Tokdemir, Ozkan Kilic, and Damla Topalli. "Performing and analyzing non-formal inspections of entity relationship diagram (ERD)." In: *Journal of Systems and Software* 86.8 (2013), pp. 2184–2195.

[S22] Gerardo Cepeda Porras and Yann-Gaël Guéhéneuc. "An empirical study on the efficiency of different design pattern representations in UML class diagrams." In: *Empirical Software Engineering* 15.5 (2010), pp. 493–522.

[S23] K R Chandrika, J Amudha, and Sithu D Sudarsan. "Recognizing eye tracking traits for source code review." In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2017, pp. 1–8. DOI: 10.1109/ETFA.2017.8247637.

[S24] Gary Cheng, Leonard KM Poon, Wilfred WF Lau, and Rachel C Zhou. "Applying Eye Tracking to Identify Students' Use of Learning Strategies in Understanding Program Code." In: *Proceedings of the 3rd International Conference on Education and Multimedia Technology*. 2019, pp. 140–144.

[S25] Natalia Chitalkina, Roman Bednarik, Marjaana Puurtinen, and Hans Gruber. "When You Ignore What You See: How to Study Proof-Readers' Error in Pseudocode Reading." In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '20 Short Papers. Stuttgart, Germany: Association for Computing Machinery, 2020. ISBN: 9781450371346. DOI: 10.1145/3379156.3391979. URL: https://doi.org/10.1145/3379156.3391979.

[S26] Ricardo Couceiro, Raul Barbosa, Joáo Duráes, Gonçalo Duarte, Joáo Castelhano, Catarina Duarte, Cesar Teixeira, Nuno Laranjeiro, Júlio Medeiros, Paulo Carvalho, Miguel Castelo Branco, and Henrique Madeira. "Spotting Problematic Code Lines using Nonintrusive Programmers' Biofeedback." In: *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. 2019, pp. 93–103. DOI: 10.1109/ISSRE.2019.00019.

[S27] Martha E Crosby, Jean Scholtz, and Susan Wiedenbeck. "The Roles Beacons Play in Comprehension for Novice and Expert Programmers." In: *PPIG*. 2002, p. 5.

[S28] Martha E Crosby and Jan Stelovsky. "How do we read algorithms? A case study." In: *Computer* 23.1 (1990), pp. 25–35.

[S29] Daniel Davis and Feng Zhu. "Understanding and improving secure coding behavior with eye tracking methodologies." In: *Proceedings of the 2020 ACM Southeast Conference*. 2020, pp. 107–114.

[S30] Hacı Ali Duru, Murat Perit Çakır, and Veysi İşler. "How does software visualization contribute to software comprehension? A grounded theory approach." In: *International Journal of Human-Computer Interaction* 29.11 (2013), pp. 743–763.

[S31] Sarah Fakhoury, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope. "The Effect of Poor Source Code Lexicon and Readability on Developers' Cognitive Load." In: *Proceedings of the 26th Conference on Program Comprehension*. ICPC '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, 286–296. ISBN: 9781450357142. DOI: 10.1145/3196321.3196347. URL: https://doi.org/10.1145/3196321.3196347.

[S32] Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. "Using psycho-physiological measures to assess task difficulty in software development." In: *Proceedings of the 36th international conference on software engineering*. 2014, pp. 402–413.

[S33] Peter Leo Gorski, Sebastian Möller, Stephan Wiefling, and Luigi Lo Iacono. ""I just looked for the solution!"On Integrating Security-Relevant Information in Non-Security API Documentation to Support Secure Coding Practices." In: *IEEE Transactions on Software Engineering* 48.9 (2022), pp. 3467–3484. DOI: 10.1109/TSE.2021.3094171.

[S34] Anurag Goswami, Gursimran Walia, Mark McCourt, and Ganesh Padmanabhan. "Using Eye Tracking to Investigate Reading Patterns and Learning Styles of Software Requirement Inspectors to Enhance Inspection Team Outcome." In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '16. Ciudad Real, Spain: Association for Computing Machinery, 2016. ISBN: 9781450344272. DOI: 10.1145/2961111.2962598. URL: https://doi.org/10.1145/2961111.2962598.

[S35] Yann-Gaël Guéhéneuc. "TAUPE: towards understanding program comprehension." In: *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*. 2006, 1–es.

[S36] Florian Hauser, Stefan Schreistter, Rebecca Reuter, Jurgen Horst Mottok, Hans Gruber, Kenneth Holmqvist, and Nick Schorr. "Code Reviews in C++ Preliminary Results from an Eye Tracking Study." In: *ACM Symposium on Eye Tracking Research and Applications*. 2020, pp. 1–5.

[S37] Prateek Hejmady and N Hari Narayanan. "Visual attention patterns during program debugging with an IDE." In: *proceedings of the symposium on eye tracking research and applications*. 2012, pp. 197–200.

[S38] Haytham Hijazi, Joao Duraes, Ricardo Couceiro, João Castelhano, Raul Barbosa, Júlio Medeiros, Miguel Castelo-Branco, Paulo de Carvalho, and Henrique Madeira. "Quality Evaluation of Modern Code Reviews Through Intelligent Biometric Program Comprehension." In: *IEEE Transactions on Software Engineering* 49.2 (2023), pp. 626–645. DOI: 10.1109/TSE.2022.3158543.

[S39] Alexander Homann, Lisa Grabinger, Florian Hauser, and Jürgen Mottok. "An Eye Tracking Study on MISRA C Coding Guidelines." In: *Proceedings of the 5th European Conference on Software Engineering Education*. ECSEE '23. Seeon/Bavaria, Germany: Association for Computing Machinery, 2023, 130–137. ISBN: 9781450399562. DOI: 10.1145/3593663.3593671. URL: https://doi.org/10.1145/3593663.3593671.

[S40] Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. "Biases and differences in code review using medical imaging and eye-tracking: genders, humans, and machines." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 456–468.

[S41] Constantina Ioannou, Per Bækgaard, Ekkart Kindler, and Barbara Weber. "Towards a tool for visualizing pupil dilation linked with source code artifacts." In: *2020 Working Conference on Software Visualization (VISSOFT)*. 2020, pp. 105–109. DOI: 10.1109/VISSOFT51673.2020.00016.

[S42] Toyomi Ishida and Hidetake Uwano. "Synchronized Analysis of Eye Movement and EEG during Program Comprehension." In: *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*. 2019, pp. 26–32. DOI: 10.1109/EMIP.2019.00012.

[S43] Ahmad Jbara and Dror G. Feitelson. "How Programmers Read Regular Code: A Controlled Experiment Using Eye Tracking." In: *2015 IEEE 23rd International Conference on Program Comprehension*. 2015, pp. 244–254. DOI: 10.1109/ICPC.2015.35.

[S44] Sebastien Jeanmart, Yann-Gael Gueheneuc, Houari Sahraoui, and Naji Habra. "Impact of the visitor pattern on program comprehension and maintenance." In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. 2009, pp. 69–78. DOI: 10.1109/ESEM.2009.5316015.

[S45] Patrick Jermann and Kshitij Sharma. "Gaze as a Proxy for Cognition and Communication." In: *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*. 2018, pp. 152–154. DOI: 10.1109/ICALT.2018.00043.

[S46] Toru Kano, Ryuichi Sakagami, and Takako Akakura. "Modeling of cognitive processes based on gaze transition during programming debugging." In: *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*. 2021, pp. 412–413. DOI: 10.1109/LifeTech52111.2021.9391940.

[S47] Philipp Kather, Rodrigo Duran, and Jan Vahrenhold. "Through (Tracking) Their Eyes: Abstraction and Complexity in Program Comprehension." In: *ACM Trans. Comput. Educ.* 22.2 (2021). DOI: 10.1145/3480171. URL: https://doi.org/10.1145/3480171.

[S48] Jozsef Katona, Attila Kovari, Cristina Costescu, Adrian Rosan, Andrea Hathazi, Ilona Heldal, Carsten Helgesen, Serge Thill, and Robert Demeter. "The Examination Task of Source-code Debugging Using GP3 Eye Tracker." In: *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 2019, pp. 329–334. DOI: 10.1109/CogInfoCom47531.2019.9089952.

[S49] Jozsef Katona, Attila Kovari, Ilona Heldal, Cristina Costescu, Adrian Rosan, Robert Demeter, Serge Thill, and Teodor Stefanut. "Using Eye- Tracking to Examine Query Syntax and Method Syntax Comprehension in LINQ." In: *2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 2020, pp. 000437–000444. DOI: 10.1109/CogInfoCom50765.2020.9237910.

[S50] Xinyu Li, Wei Liu, Huitong Liu, Jing Xu, and Wenqing Cheng. "Task-oriented Analysis on Debugging Process Based on Eye Movements and IDE Interactions." In: *2021 16th International Conference on Computer Science and Education (ICCSE)*. 2021, pp. 379–384. DOI: 10.1109/ICCSE51940.2021.9569438.

[S51] Xinyu Li, Wei Liu, Weiwei Wang, Jinrong Zhong, and Menglin Yu. "Assessing Students' Behavior in Error Finding Programming Tests: An Eye-Tracking Based Approach." In: *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*. 2019, pp. 1–6. DOI: 10.1109/TALE48000.2019.9225906.

[S52] Calvin Liang, Jakob Karolus, Thomas Kosch, and Albrecht Schmidt. "On the suitability of real-time assessment of programming proficiency using gaze properties." In: *Proceedings of the 7th ACM International Symposium on Pervasive Displays*. 2018, pp. 1–2.

[S53] Yu-Tzu Lin, Yi-Zhi Liao, Xiao Hu, and Cheng-Chih Wu. "EEG Activities During Program Comprehension: An Exploration of Cognition." In: *IEEE Access* 9 (2021), pp. 120407–120421. DOI: 10.1109/ACCESS.2021.3107795.

[S54] Yu-Tzu Lin, Cheng-Chih Wu, Ting-Yun Hou, Yu-Chih Lin, Fang-Ying Yang, and Chia-Hu Chang. "Tracking Students' Cognitive Processes During Program Debugging—An Eye-Movement Approach." In: *IEEE Transactions on Education* 59.3 (2016), pp. 175–186. DOI: 10.1109/TE.2015.2487341.

[S55] Lianzhen Liu, Wei Liu, Xinyu Li, Jing Xu, and Wenqing Cheng. "An Analysis Scheme to Interpret Students' Cognitive Process in Error Finding Test." In: *Proceedings of the 2nd World Symposium on Software Engineering*. WSSE '20. Chengdu, China: Association for Computing Machinery, 2020, 220–225. ISBN: 9781450387873. DOI: 10.1145/3425329.3425350. URL: https://doi.org/10.1145/3425329.3425350.

[S56] Ian McChesney and Raymond Bond. "Observations on the Linear Order of Program Code Reading Patterns in Programmers with Dyslexia." In: *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*. EASE '20. Trondheim, Norway: Association for Computing Machinery, 2020, 81–89. ISBN: 9781450377317. DOI: 10.1145/3383219.3383228. URL: https://doi.org/10.1145/3383219.3383228.

[S57] Ian McChesney and Raymond Bond. "Eye Tracking Analysis of Code Layout, Crowding and Dyslexia - An Open Data Set." In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '21 Short Papers. Virtual Event, Germany: Association for Computing Machinery, 2021. ISBN: 9781450383455. DOI: 10.1145/3448018.3457420. URL: https://doi.org/10.1145/3448018.3457420.

[S58] Jean Melo, Fabricio Batista Narcizo, Dan Witzner Hansen, Claus Brabrand, and Andrzej Wasowski. "Variability through the eyes of the programmer." In: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE. 2017, pp. 34–44.

[S59] Markus Nivala, Florian Hauser, Jürgen Mottok, and Hans Gruber. "Developing visual expertise in software engineering: An eye tracking study." In: *2016 IEEE Global Engineering Education Conference (EDUCON)*. 2016, pp. 613–620. DOI: 10.1109/EDUCON.2016.7474614.

[S60] Unaizah Obaidellah, Tanja Blascheck, Drew T. Guarnera, and Jonathan Maletic. "A Fine-Grained Assessment on Novice Programmers' Gaze Patterns on Pseudocode Problems." In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '20 Short Papers. Stuttgart, Germany: Association for Computing Machinery, 2020. ISBN: 9781450371346. DOI: 10.1145/3379156.3391982. URL: https://doi.org/10.1145/3379156.3391982.

[S61] Unaizah Obaidellah and Mohammed Al Haek. "Evaluating Gender Difference on Algorithmic Problems Using Eye-Tracker." In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ETRA '18. Warsaw, Poland: Association for Computing Machinery, 2018. ISBN: 9781450357067. DOI: 10.1145/3204493.3204537. URL: https://doi.org/10.1145/3204493.3204537.

[S62] Unaizah Obaidellah, Michael Raschke, and Tanja Blascheck. "Classification of strategies for solving programming problems using AoI sequence analysis." In: *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. 2019, pp. 1–9.

[S63] Benedito de Oliveira, Márcio Ribeiro, José Aldo Silva da Costa, Rohit Gheyi, Guilherme Amaral, Rafael de Mello, Anderson Oliveira, Alessandro Garcia, Rodrigo Bonifácio, and Baldoino Fonseca. "Atoms of confusion: The eyes do not lie." In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 2020, pp. 243–252.

[S64] Sofia Papavlasopoulou, Kshitij Sharma, Michail Giannakos, and Letizia Jaccheri. "Using Eye-Tracking to Unveil Differences Between Kids and Teens in Coding Activities." In: *Proceedings of the 2017 Conference on Interaction Design and Children*. IDC '17. Stanford, California, USA: Association for Computing Machinery, 2017, 171–181. ISBN: 9781450349215. DOI: 10.1145/3078072.3079740. URL: https://doi.org/10.1145/3078072.3079740.

[S65] Kang-il Park and Bonita Sharif. "Assessing Perceived Sentiment in Pull Requests with Emoji: Evidence from Tools and Developer Eye Movements." In: *2021 IEEE/ACM Sixth International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. 2021, pp. 1–6. DOI: 10.1109/SEmotion52567.2021.00009.

[S66] Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. "Correlates of Programmer Efficacy and Their Link to Experience: A Combined EEG and Eye-Tracking Study." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. Singapore, Singapore: Association for Computing Machinery, 2022, 120–131. ISBN: 9781450394130. DOI: 10.1145/3540250.3549084. URL: https://doi.org/10.1145/3540250.3549084.

[S67] Norman Peitek, Janet Siegmund, and Sven Apel. "What drives the reading order of programmers." In: *An eye tracking study. In ICPC* (2020).

[S68] Norman Peitek, Janet Siegmund, Chris Parnin, Sven Apel, and André Brechmann. "Toward Conjoint Analysis of Simultaneous Eye-Tracking and FMRI Data for Program-Comprehension Studies." In: *Proceedings of the Workshop on Eye Movements in Programming*. EMIP '18. Warsaw, Poland: Association for Computing Machinery, 2018. ISBN:

9781450357920. DOI: 10.1145/3216723.3216725. URL: https://doi.org/10.1145/3216723.3216725.

[S69] Fei Peng, Chunyu Li, Xiaohan Song, Wei Hu, and Guihuan Feng. "An Eye Tracking Research on Debugging Strategies towards Different Types of Bugs." In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. 2016, pp. 130–134. DOI: 10.1109/COMPSAC.2016.57.

[S70] Cole S. Peterson. "Investigating the Effect of Polyglot Programming on Developers." In: *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2021, pp. 1–2. DOI: 10.1109/VL/HCC51201.2021.9576404.

[S71] Cole S. Peterson, Kang-il Park, Isaac Baysinger, and Bonita Sharif. "An Eye Tracking Perspective on How Developers Rate Source Code Readability Rules." In: *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. 2021, pp. 138–139. DOI: 10.1109/ASEW52652.2021.00037.

[S72] Cole S. Peterson, Jonathan A. Saddler, Tanja Blascheck, and Bonita Sharif. "Visually Analyzing Students' Gaze on C++ Code Snippets." In: *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*. 2019, pp. 18–25. DOI: 10.1109/EMIP.2019.00011.

[S73] Cole S. Peterson, Jonathan A. Saddler, Natalie M. Halavick, and Bonita Sharif. "A Gaze-Based Exploratory Study on the Information Seeking Behavior of Developers on Stack Overflow." In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI EA '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, 1–6. ISBN: 9781450359719. DOI: 10.1145/3290607.3312801. URL: https://doi.org/10.1145/3290607.3312801.

[S74] Razvan Petrusel and Jan Mendling. "Eye-tracking the factors of process model comprehension tasks." In: *Advanced Information Systems Engineering: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings 25*. Springer. 2013, pp. 224–239.

[S75] Paige Rodeghero, Cheng Liu, Paul W. McBurney, and Collin McMillan. "An Eye-Tracking Study of Java Programmers and Application to Source Code Summarization." In: *IEEE Transactions on Software Engineering* 41.11 (2015), pp. 1038–1054. DOI: 10.1109/TSE.2015.2442238.

[S76] Paige Rodeghero, Collin McMillan, Paul W McBurney, Nigel Bosch, and Sidney D'Mello. "Improving automated source code summarization via an eye-tracking study of programmers." In: *Proceedings of the 36th international conference on Software engineering*. 2014, pp. 390–401.

[S77] Jonathan A. Saddler, Cole S. Peterson, Sanjana Sama, Shruthi Nagaraj, Olga Baysal, Latifa Guerrouj, and Bonita Sharif. "Studying Developer Reading Behavior on Stack Overflow during API Summarization Tasks." In: *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2020, pp. 195–205. DOI: 10.1109/SANER48275.2020.9054848.

[S78] Djan Santos and Cláudio Sant' Anna. "How Does Feature Dependency Affect Configurable System Comprehensibility?" In: *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. 2019, pp. 19–29. DOI: 10.1109/ICPC.2019.00016.

[S79] Zohreh Sharafi, Ian Bertram, Michael Flanagan, and Westley Weimer. "Eyes on Code: A Study on Developers' Code Navigation Strategies." In: *IEEE Transactions on Software Engineering* 48.5 (2022), pp. 1692–1704. DOI: 10.1109/TSE.2020.3032064.

[S80] Zohreh Sharafi, Yu Huang, Kevin Leach, and Westley Weimer. "Toward an objective measure of developers' cognitive activities." In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30.3 (2021), pp. 1–40.

[S81] Zohreh Sharafi, Alessandro Marchetto, Angelo Susi, Giuliano Antoniol, and Yann-Gaël Guéhéneuc. "An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension." In: *2013 21st International Conference on Program Comprehension (ICPC)*. IEEE. 2013, pp. 33–42.

[S82] Zohreh Sharafi, Zéphyrin Soh, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. "Women and men—different but equal: On the impact of identifier style on source code reading." In: *2012 20th IEEE International Conference on Program Comprehension (ICPC)*. IEEE. 2012, pp. 27–36.

[S83] Bonita Sharif, Michael Falcone, and Jonathan I Maletic. "An eye-tracking study on the role of scan time in finding source code defects." In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. 2012, pp. 381–384.

[S84] Bonita Sharif, Grace Jetty, Jairo Aponte, and Esteban Parra. "An empirical study assessing the effect of seeit 3d on comprehension." In: *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE. 2013, pp. 1–10.

[S85] Bonita Sharif and Jonathan I Maletic. "An eye tracking study on the effects of layout in understanding the role of design patterns." In: *2010 IEEE International Conference on Software Maintenance*. IEEE. 2010, pp. 1–10.

[S86] Kshitij Sharma, Patrick Jermann, Marc-Antoine Nüssli, and Pierre Dillenbourg. "Understanding collaborative program comprehension: Interlacing gaze and dialogues." In: (2013).

[S87] Georg Simhandl, Philipp Paulweber, and Uwe Zdun. "Design of an executable specification language using eye tracking." In: *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*. IEEE. 2019, pp. 37–40.

[S88] Zéphyrin Soh, Zohreh Sharafi, Bertrand Van den Plas, Gerardo Cepeda Porras, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. "Professional status and expertise for UML class diagram comprehension: An empirical study." In: *2012 20th IEEE International Conference on Program Comprehension (ICPC)*. IEEE. 2012, pp. 163–172.

[S89] Louis Spinelli, Maulishree Pandey, and Steve Oney. "Attention patterns for code animations: using eye trackers to evaluate dynamic code presentation techniques." In: *Companion Proceedings of the 2nd International Conference on the Art, Science, and Engineering of Programming*. 2018, pp. 99–104.

[S90] Randy Stein and Susan E Brennan. "Another person's eye gaze as a cue in solving programming problems." In: *Proceedings of the 6th international conference on Multimodal interfaces*. 2004, pp. 9–15.

[S91] Renske Talsma, Erik Barendsen, and Sjaak Smetsers. "Analyzing the influence of block highlighting on beginning programmers' reading behavior using eye tracking." In: *Proceedings of the 9th Computer Science Education Research Conference*. 2020, pp. 1–10.

[S92] Rachel Turner, Michael Falcone, Bonita Sharif, and Alina Lazar. "An eye-tracking study assessing the comprehension of C++ and Python source code." In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. 2014, pp. 231–234.

[S93] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. "Analyzing individual performance of source code review using reviewers' eye movement." In: *Proceedings of the 2006 symposium on Eye tracking research & applications*. 2006, pp. 133–140.

[S94] Maureen Villamor and Ma. Mercedes Rodrigo. "Predicting Successful Collaboration in a Pair Programming Eye Tracking Experiment." In: *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. UMAP '18. Singapore, Singapore: Association for Computing Machinery, 2018, 263–268. ISBN: 9781450357845. DOI: 10.1145/3213586.3225234. URL: https://doi.org/10.1145/3213586.3225234.

[S95] Braden Walters, Timothy Shaffer, Bonita Sharif, and Huzefa Kagdi. "Capturing software traceability links from developers' eye gazes." In: *Proceedings of the 22nd International Conference on Program Comprehension*. 2014, pp. 201–204.

[S96] Thomas Weber, Christina Winiker, and Heinrich Hussmann. "A Closer Look at Machine Learning Code." In: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI EA '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380959. DOI: 10.1145/3411763.3451679. URL: https://doi.org/10.1145/3411763.3451679.

[S97] Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan I Maletic. "Assessing the comprehension of UML class diagrams via eye tracking." In: *15th IEEE International Conference on Program Comprehension (ICPC'07)*. IEEE. 2007, pp. 113–122.

A.2 APPENDIX B: EXTRACTION FORM

Table A.1: Extraction form

Data item	Description	RQs
Title	Title of the study	
ID	ID of the study (S1, S2, . . .)	
DOI	Document object identifier to link to the concrete instance of a paper. If does not exist, other link to concrete instance of a paper.	
citation (in APA)	Paper citation in APA format.	
year	Publication year of the paper.	
venue	Either journal, conference, or workshop.	
tool	Tool that is used to analyze the eye tracking data and which part of the eye tracking data, if named. If a custom tool is used, a description of the tool used will be added.	1
tool justification/consequence	Justifications and/or consequences that the author gave for the choice of tool to analyze the eye tracking data, if given.	2
eyetracker	Which eye tracker the study was using to measure the participants.	1
configuration eyetracker	Which configuration the eye tracker has, for example: Distance between screen and participant, size/resolution of screen,. . .	1
study task	Which task the participants needed to perform to complete the study. (program comprehension, debugging, Traceability, Collaborative, Comprehension(non-code),. . .)	1
#participants	Participant sample size.	1
demographics	Composition of the sample, e.g. professionals, faculty, students.	1
#items per participant	Number of items a participant had to look at.	1
item type	What kind of item the participant had to look at, for example for diagrams it would be a diagram, for code comprehension a code snippet as a list	1
item language	Which programming language the item originates from, if it is possible to describe, as a list	1
item scrolling	If the items were big enough that scrolling was involved to see all of it during the eye tracking measurements	1
replication package	Does the study have a replication package available	3

A.3 APPENDIX C: RESULTS OF RQ3

Table A.2: Total fixation duration in milliseconds by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	EMIP	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	EMIP / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	275307	41766.7	605255	480676	57.27%	8.69%	125.92%
P114	63486	7058.3	170860	147388	43.07%	4.79%	115.93%
P131	83701	7400	240537	191336	43.75%	3.87%	125.71%
P157	227271	53475	393077	348104	65.29%	15.36%	112.92%
P214	321484	104575	531027	424945	75.65%	24.61%	124.96%
P237	205723	20141.7	513272	370787	55.48%	5.43%	138.43%
P267	227000	56441.7	418484	366252	61.98%	15.41%	114.26%
P270	355742	92908.3	633489	536216	66.34%	17.33%	118.14%
P316	19185	958.3	124078	76006	25.24%	1.26%	163.25%
P323	68977	9716.7	178955	143876	47.94%	6.75%	124.38%
P365	278466	68216.7	401796	362687	76.78%	18.81%	110.78%
P370	199139	51941.7	305843	256845	77.53%	20.22%	119.08%
P402	327360	94800.0	519945	421034	77.75%	22.52%	123.49%
P450	94024	11783.3	221153	185581	50.66%	6.35%	119.17%
P459	101069	11316.7	379203	206081	49.04%	5.49%	184.01%
P469	123992	13308.3	325529	289593	42.82%	4.60%	112.41%
P513	150233	34475	260499	218014	68.91%	15.81%	119.49%
P536	109585	16608.3	280028	225783	48.54%	7.36%	124.03%
P561	89493	7216.7	321156	237171	37.73%	3.04%	135.41%
P611	209808	50491.7	305412	268719	78.08%	18.79%	113.65%
P642	106427	10725	259606	220171	48.34%	4.87%	117.91%
P645	125351	22100	317457	289072	43.36%	7.65%	109.82%
P653	48119	3050	235771	155650	30.91%	1.96%	151.48%
P708	172018	51075	284344	244048	70.49%	20.93%	116.51%
P751	400894	119016.7	619912	558417	71.79%	21.31%	111.01%
P758	85093	11300	291354	215887	39.42%	5.23%	134.96%
P812	207169	30475	391232	313256	66.13%	9.73%	124.89%
P819	343749	153741.7	472490	394164	87.21%	39.00%	119.87%
P842	199931	24983.3	375159	307517	65.01%	8.12%	122.00%
P900	330268	100000	476772	416780	79.24%	23.99%	114.39%
Average Similarity to Tobii Pro Lab					**58.39%**	**12.31%**	**124.94%**
Average Variance					**14.15%**	**7.60%**	**10.64%**

Table A.3: Average fixation duration in milliseconds

Table A.4: Average fixation duration in milliseconds by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	EMIP	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	EMIP / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	94.67	68.25	213.04	54.00	175.31%	126.38%	394.51%
P114	85.79	64.17	208.62	52.51	163.39%	122.21%	397.32%
P131	87.46	65.49	199.12	37.84	231.16%	173.08%	526.27%
P157	108.17	71.88	216.10	88.40	122.37%	81.31%	244.46%
P214	121.73	77.06	264.19	87.42	139.25%	88.15%	302.21%
P237	88.03	66.04	204.74	44.47	197.95%	148.50%	460.40%
P267	111.33	73.49	240.78	88.04	126.45%	83.47%	273.49%
P270	117.37	75.97	232.13	75.64	155.17%	100.43%	306.89%
P316	70.53	63.89	264.56	20.12	350.50%	317.49%	1314.68%
P323	91.12	69.40	216.13	41.62	218.94%	166.76%	519.31%
P365	119.82	71.66	253.34	115.76	103.51%	61.90%	218.84%
P370	123.15	71.84	247.85	105.92	116.28%	67.83%	234.00%
P402	115.72	74.70	221.44	67.33	171.86%	110.95%	328.87%
P450	90.58	65.46	194.68	51.07	177.38%	128.19%	381.21%
P459	86.09	67.76	287.28	23.35	368.74%	290.25%	1230.48%
P469	89.14	66.54	235.04	61.96	143.87%	107.40%	379.35%
P513	107.23	72.27	205.44	80.87	132.61%	89.38%	254.05%
P536	98.02	70.08	271.87	50.19	195.31%	139.64%	541.74%
P561	75.97	63.30	190.71	28.62	265.45%	221.19%	666.36%
P611	127.23	69.55	260.81	109.24	116.48%	63.67%	238.76%
P642	90.89	66.20	252.78	69.50	130.77%	95.26%	363.72%
P645	94.60	68.85	220.46	53.71	176.14%	128.18%	410.45%
P653	76.14	64.89	247.66	26.31	289.34%	246.61%	941.15%
P708	124.92	74.34	258.03	80.73	154.74%	92.09%	319.61%
P751	117.53	75.04	225.83	104.65	112.31%	71.71%	215.80%
P758	86.13	66.86	267.54	35.19	244.75%	190.01%	760.29%
P812	100.71	68.18	231.77	62.07	162.26%	109.84%	373.42%
P819	140.71	83.06	218.85	94.21	149.36%	88.17%	232.30%
P842	97.43	66.45	204.67	72.58	134.24%	91.55%	282.00%
P900	128.66	75.59	252.13	103.63	124.16%	72.94%	243.31%
Average Similarity to Tobii Pro Lab					**178.33%**	**129.15%**	**445.18%**
Average Variance					**50.48%**	**48.75%**	**196.94%**

Table A.5: Calculated saccades counts by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	1174	2840	8901	13.19%	31.91%
P114	309	819	2807	11.01%	29.18%
P131	291	1208	5057	5.75%	23.89%
P157	337	1819	3938	8.56%	46.19%
P214	680	2010	4861	13.99%	41.35%
P237	548	2507	8338	6.57%	30.07%
P267	415	1738	4160	9.98%	41.78%
P270	854	2729	7089	12.05%	38.50%
P316	157	469	3777	4.16%	12.42%
P323	350	828	3457	10.12%	23.95%
P365	412	1586	3133	13.15%	50.62%
P370	335	1234	2425	13.81%	50.89%
P402	743	2348	6253	11.88%	37.55%
P450	230	1136	3634	6.33%	31.26%
P459	705	1320	8827	7.99%	14.95%
P469	376	1385	4674	8.04%	29.63%
P513	243	1268	2696	9.01%	47.03%
P536	361	1030	4499	8.02%	22.89%
P561	484	1684	8287	5.84%	20.32%
P611	279	1171	2460	11.34%	47.60%
P642	231	1027	3168	7.29%	32.42%
P645	310	1440	5382	5.76%	26.76%
P653	276	952	5915	4.67%	16.09%
P708	356	1102	3023	11.78%	36.45%
P751	600	2745	5336	11.24%	51.44%
P758	390	1089	6135	6.36%	17.75%
P812	350	1688	5047	6.93%	33.45%
P819	806	2159	4184	19.26%	51.60%
P842	326	1833	4237	7.69%	43.26%
P900	600	1891	4022	14.92%	47.02%
Average Similarity to Tobii Pro Lab				**9.56%**	**34.27%**
Average Variance				**2.93%**	**10.10%**

Table A.6: Total saccade duration in milliseconds by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	921514	329092	80494	1144.82%	408.84%
P114	313829	147017	24469	1282.56%	600.83%
P131	483984	246438	38120	1269.63%	646.48%
P157	544167	156562	52390	1038.68%	298.84%
P214	749440	225648	57396	1305.74%	393.14%
P237	782713	278835	69037	1133.76%	403.89%
P267	569643	156361	52288	1089.43%	299.04%
P270	899442	276474	77130	1166.14%	358.45%
P316	424430	277348	15899	2669.54%	1744.44%
P323	377388	204446	23849	1582.41%	857.25%
P365	500691	102392	51587	970.58%	198.48%
P370	411428	110385	38151	1078.42%	289.34%
P402	738307	224676	74800	987.04%	300.37%
P450	363368	144217	35569	1021.59%	405.46%
P459	799317	432091	38687	2066.11%	1116.89%
P469	513002	193633	39139	1310.72%	494.73%
P513	352738	95013	39161	900.74%	242.62%
P536	481634	206083	33218	1449.92%	620.40%
P561	682863	367973	46536	1467.39%	790.73%
P611	417410	118961	41457	1006.85%	286.95%
P642	390305	132267	33140	1177.75%	399.12%
P645	530188	215417	49770	1065.28%	432.82%
P653	489824	258381	30742	1593.34%	840.48%
P708	411093	130834	32911	1249.11%	397.54%
P751	784188	170201	84649	926.40%	201.07%
P758	557110	266953	36475	1527.37%	731.88%
P812	531087	149404	50883	1043.74%	293.62%
P819	675741	211519	60534	1116.30%	349.42%
P842	510194	139237	54252	940.42%	256.65%
P900	627856	156422	60829	1032.17%	257.15%
Average Similarity to Tobii Pro Lab				**1253.80%**	**497.23%**
Average Variance				**248.86%**	**231.62%**

Table A.7: Average saccade duration in milliseconds by software tools using the data of McChesney et al. [13] with Tobii Pro Lab as the originaly used software tool

Participant	PyGaze	Ogama	Tobii Pro Lab	PyGaze / Tobii Pro Lab	Ogama / Tobii Pro Lab
P100	785.60	115.88	27.10	2898.66%	427.55%
P114	1018.93	179.51	26.20	3889.31%	685.20%
P131	1663.18	204.00	28.64	5807.15%	712.30%
P157	1619.54	86.07	27.59	5870.42%	311.98%
P214	1103.74	112.26	26.35	4188.35%	426.00%
P237	1430.92	111.22	26.60	5378.62%	418.07%
P267	1375.95	89.97	27.59	4986.66%	326.05%
P270	1054.45	101.31	25.94	4064.39%	390.50%
P316	2720.71	591.36	25.28	10763.72%	2339.55%
P323	1081.34	246.92	26.35	4103.37%	936.97%
P365	1215.27	64.56	29.80	4077.83%	216.63%
P370	1231.82	89.45	29.95	4113.49%	298.72%
P402	995.02	95.69	30.16	3299.01%	317.26%
P450	1579.86	126.95	29.57	5343.34%	429.37%
P459	1135.39	327.34	26.19	4334.73%	1249.73%
P469	1368.01	139.81	23.55	5809.10%	593.68%
P513	1457.60	74.93	31.23	4667.46%	239.94%
P536	1337.87	200.08	25.73	5199.57%	777.60%
P561	1413.80	218.51	26.40	5356.11%	827.82%
P611	1501.47	101.59	32.21	4661.21%	315.38%
P642	1689.63	128.79	26.36	6408.77%	488.50%
P645	1715.82	149.60	27.99	6129.65%	534.42%
P653	1781.18	271.41	25.64	6946.95%	1058.55%
P708	1158.01	118.72	27.77	4169.55%	427.48%
P751	1309.16	62.00	29.38	4455.69%	211.03%
P758	1428.49	245.14	26.00	5494.63%	942.91%
P812	1521.74	88.51	26.97	5643.38%	328.24%
P819	839.43	97.97	28.03	2995.29%	349.58%
P842	1569.83	75.96	28.67	5474.66%	264.91%
P900	1048.17	82.72	29.40	3565.19%	281.36%
Average Similarity to Tobii Pro Lab				**5003.21%**	**570.91%**
Average Variance				**1038.74%**	**294.35%**

BIBLIOGRAPHY

[1] Richard Andersson, Linnea Larsson, Kenneth Holmqvist, Martin Stridh, and Marcus Nyström. "One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms." In: *Behavior research methods* 49 (2017), pp. 616–637.

[2] John Bailey, Cheng Zhang, David Budgen, Mark Turner, and Stuart Charters. "Search Engine Overlaps : Do they agree or disagree?" In: *Second International Workshop on Realising Evidence-Based Software Engineering (REBSE '07)*. 2007, pp. 2–2. DOI: 10.1109/REBSE.2007.4.

[3] Daniela S Cruzes and Tore Dyba. "Recommended steps for thematic synthesis in software engineering." In: *2011 international symposium on empirical software engineering and measurement*. IEEE. 2011, pp. 275–284.

[4] Benoît De Smet, Lorent Lempereur, Zohreh Sharafi, Yann-Gaël Guéhéneuc, Giuliano Antoniol, and Naji Habra. "Taupe: Visualizing and analyzing eye-tracking data." In: *Science of computer programming* 79 (2014), pp. 260–278.

[5] Andrew T Duchowski. *Eye tracking methodology: Theory and practice*. Springer, 2017.

[6] Joseph H Goldberg and Xerxes P Kotval. "Computer interface evaluation using eye movements: methods and constructs." In: *International journal of industrial ergonomics* 24.6 (1999), pp. 631–645.

[7] Drew T Guarnera, Corey A Bryant, Ashwin Mishra, Jonathan I Maletic, and Bonita Sharif. "itrace: Eye tracking infrastructure for development environments." In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 2018, pp. 1–3.

[8] Kenneth Holmqvist, Saga Lee Örbom, Ignace TC Hooge, Diederick C Niehorster, Robert G Alexander, Richard Andersson, Jeroen S Benjamins, Pieter Blignaut, Anne-Marie Brouwer, Lewis L Chuang, et al. "Eye tracking: empirical foundations for a minimal reporting guideline." In: *Behavior research methods* 55.1 (2023), pp. 364–416.

[9] Marcel A Just and Patricia A Carpenter. "A theory of reading: from eye fixations to comprehension." In: *Psychological review* 87.4 (1980), p. 329.

[10] Staffs Keele et al. *Guidelines for performing systematic literature reviews in software engineering*. 2007.

[11] Oleg V Komogortsev, Denise V Gobert, Sampath Jayarathna, Sandeep M Gowda, et al. "Standardization of automated analyses of oculomotor fixation and saccadic behaviors." In: *IEEE Transactions on biomedical engineering* 57.11 (2010), pp. 2635–2645.

[12] Linnéa Larsson, Marcus Nyström, and Martin Stridh. "Detection of saccades and postsaccadic oscillations in the presence of smooth pursuit." In: *IEEE Transactions on biomedical engineering* 60.9 (2013), pp. 2484–2493.

[13] Ian McChesney and Raymond Bond. "Eye tracking analysis of code layout, crowding and dyslexia-an open data set." In: *ACM Symposium on Eye Tracking Research and Applications*. 2021, pp. 1–6.

[14] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. "Systematic mapping studies in software engineering." In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. 2008, pp. 1–10.

[15] A Poole and Linden Ball. "Eye tracking in human-computer interaction and usability research: Current status and future prospects." In: Jan. 2006, pp. 211–219.

[16] Dario D Salvucci and Joseph H Goldberg. "Identifying fixations and saccades in eye-tracking protocols." In: *Proceedings of the 2000 symposium on Eye tracking research & applications*. 2000, pp. 71–78.

[17] Zohreh Sharafi, Yu Huang, Kevin Leach, and Westley Weimer. "Toward an objective measure of developers' cognitive activities." In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30.3 (2021), pp. 1–40.

[18] Zohreh Sharafi, Timothy Shaffer, Bonita Sharif, and Yann-Gaël Guéhéneuc. "Eye-tracking metrics in software engineering." In: *2015 Asia-Pacific Software Engineering Conference (APSEC)*. IEEE. 2015, pp. 96–103.

[19] Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. "A practical guide on conducting eye tracking studies in software engineering." In: *Empirical Software Engineering* 25 (2020), pp. 3128–3174.

[20] 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.

[21] Bonita Sharif and Huzefa Kagdi. "On the use of eye tracking in software traceability." In: *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*. 2011, pp. 67–70.

[22] Heino Widdel. "Operational problems in analysing eye movements." In: *Advances in psychology*. Vol. 22. Elsevier, 1984, pp. 21–29.

[23] Vlas Zyrianov, Cole S Peterson, Drew T Guarnera, Joshua Behler, Praxis Weston, Bonita Sharif, and Jonathan I Maletic. "Deja Vu: semantics-aware recording and replay of high-speed eye tracking and interaction data to support cognitive studies of software engineering tasks—methodology and analyses." In: *Empirical software engineering* 27.7 (2022), p. 168.