Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Andreas Rimolsrønning and Ola Plassen

Eye Tracking Studio – Designing and Evaluating a Feedback System Utilizing Eye-Tracking to Improve Remote Collaboration Between Pair Programmers

Master's thesis in Informatics
Supervisor: Kshitij Sharma
June 2022

NTNU
Norwegian University of Science and Technology

Andreas Rimolsrønning and Ola Plassen

Eye Tracking Studio – Designing and Evaluating a Feedback System Utilizing Eye-Tracking to Improve Remote Collaboration Between Pair Programmers

Master's thesis in Informatics
Supervisor: Kshitij Sharma
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Eye Tracking Studio – Designing and Evaluating a Feedback System Utilizing Eye-Tracking to Improve Remote Collaboration Between Pair Programmers

Andreas Rimolsrønning Ola Plassen

June 9, 2022

Abstract

Remote pair programming has in recent years seen a surge in popularity, especially with the work from home movement. However, remote collaboration does not carry all the benefits of co-located collaboration. There is a lack of non-verbal cues, and referential communication is more difficult with remote work. These disadvantages affect the quality of collaboration and the cognitive load of the remote pair programmers. With the help of eye trackers, we design and evaluate a feedback system for remediating the disadvantages of remote pair programming. The system is implemented as a Visual Studio Code extension which provides real-time gaze awareness and cognitive-load-based feedback. The feedback system visualizes pair programmers' gazes in the source code, and it indicates when they are looking at the same area. It also provides feedback when their cognitive load is too high. We conducted a within-subject-design experiment to test and evaluate the feedback system. Pair programmers completed two debugging tasks in a simulated remote pair programming environment with and without the feedback system. The feedback system helps remote pair programmers manage their focus with their partners in a non-distracting manner. Programmers spent more time looking at the same code area, i.e., the system encourages and increases joint focus. However, the system does not reduce the programmers' overall cognitive load. The cognitive-load-based feedback was intrusive, although helpful. Future work should investigate how cognitive-load-based alerts can be better triggered and how to deliver the feedback.

Sammendrag

Fjernparprogrammering har de siste årene sett en økning i popularitet, spesielt på grunn av økt bruk av hjemmekontor. Fjernsamarbeid har imidlertid ikke alle fordelene som samlokalisert samarbeid. Det er mangel på ikke-verbale signaler, og referansekommunikasjon er vanskeligere med fjernarbeid. Disse ulempene påvirker kvaliteten på samarbeidet og den kognitive belastningen til fjernparprogrammererne. Ved hjelp av øyesporere har vi designet og evaluert et tilbakemeldingssystem for å redusere ulempene med fjernparprogrammering. Systemet er implementert som en Visual Studio Code-utvidelse som gir blikkbevissthet og kognitive belastningsbaserte tilbakemeldinger i sanntid. Tilbakemeldingssystemet visualiserer parprogrammerernes blikk i kildekoden, og indikerer når de ser på det samme området. Systemet gir også tilbakemelding når deres kognitive belastning er for høy. Vi gjennomførte et innen-emne-designeksperiment for å teste og evaluere tilbakemeldingssystemet. Parprogrammererne fullførte to feilsøkingsoppgaver i et simulert fjernparprogrammeringsmiljø med og uten hjelp fra tilbakemeldingssystemet. Tilbakemeldingssystemet hjelper fjernparprogrammerere med å koordinere fokus med partnerne sine på en ikke-distraherende måte. Programmerere brukte mer tid på å se på det samme kodeområdet. Det vil si at systemet fremmer økt felles fokus. Systemet reduserer imidlertid ikke programmerernes kognitive belastning. Den kognitive belastningsbaserte tilbakemeldingen var påtrengende, men nyttig. Fremtidig arbeid bør undersøke hvordan kognitive belastningsbaserte tilbakemeldinger kan bedre utløses og hvordan man kan vise frem tilbakemeldingene.

Acknowledgment

First, we would like to extend our sincere thanks to our thesis advisor Kshitij Sharma of the Department of Computer Science at NTNU. His support and guidance throughout these two semesters have been invaluable. He allowed this paper to be our own work but provided directions and feedback whenever we needed it. We would also like to thank a fellow eye-tracking master's student Fredrik Svendsen for helping us with the recruitment process for our experiment. We are also thankful to Sigrid Marita Kvamme and Espen Gudmundsen for keeping us company in much-needed lunch breaks and late-night writing sessions. We want to thank Monica Plassen and Dee Roberts for their help with proofreading and for giving vital feedback on our thesis. Lastly, we want to extend our gratitude to our family and friends for continuous encouragement throughout our years of study.

Contents

Figures

Tables

Acronyms

API application programming interface. 20, 21, 30, 36, 39

CAQ Computer Attitude Questionnaire. 26, 47

CLT Cognitive load theory. 2, 8

CSCW Computer Supported Cooperative Work. 3

ETS Eye Tracking Studio. 21, 25, 41, 43, 63, 66, 72

FC feedback condition. 26, 28, 44, 45, 47, 48, 54, 59, 63–66, 68

IP Internet Protocol address. 31, 34

IPA Index of Pupillary Activity. 10, 11, 28, 30, 39, 40, 44, 48, 65

JVA joint visual attention. 1, 3, 4, 8, 11, 14, 16, 21, 26, 31, 39, 41, 44, 45, 49–53, 60, 63, 66, 67, 70, 71

LAN Local Area Network. 43

NFC no-feedback condition. 26, 28, 44, 45, 47, 59, 63–66, 68

NTNU Norwegian University of Science and Technology. 25, 69

PP pair programming. 1, 10

RPP remote pair programming. 1–4, 8, 12, 17, 25, 26, 30, 34, 43, 63–65, 69–71

VPN Virtual Private Network. 43

VS Code Visual Studio Code. 4, 20, 21, 25, 30, 31, 34, 36, 39, 41, 43, 68, 71

Chapter 1

Introduction

1.1 Motivation

Remote pair programming (RPP) is increasing in popularity in conjunction with improvements in technology. Recently, due to the COVID-19 pandemic, there has been a surge in programmers working remotely from their homes [1]. However, RPP does not carry all the benefits of co-located pair programming (PP). Referential communication, which occurs when speakers refer to an entity and its location, can be challenging because there is a lack of non-verbal cues in remote collaboration, such as gestures and pointing at a screen. This referring is often carried out verbally using deictics and can cause misinterpretation in a remote setting because of ambiguous verbal communication. However, gaze could be a natural source of input information that can aid with disambiguation and benefit collaboration [2].

Therefore, when pair programmers work remotely, communicating location in code is not as easy as with co-located work. Ineffective communication can make it difficult for pair programmers to know where their partner is looking, align their attention and know whether or not they are on the same page. Furthermore, it causes an unnecessary increase in cognitive load, which affects the pair's performance when programming [3]. Tsai *et al.* [4] found that distributed pair programmers had a significantly higher cognitive load than co-located pair programmers. According to Pietinen *et al.* [5] the frequency of the partners looking at the same area, or their joint visual attention (JVA), in a collaborative programming context, can indicate the efficiency of their collaboration. Thus, the collaborating pair must be in a similar visual and mental space.

Therefore, we propose a feedback system for remote pair programmers which will alleviate the aforementioned issues related to RPP. We will also evaluate how

the system's feedback can improve collaboration. We are not aware of any existing feedback systems for helping pair programmers achieve joint focus and reduce cognitive load.

Eye tracking

An eye tracker is used to measure and record the user's gaze. Eye trackers have evolved to become affordable and versatile devices and ubiquitous.

The gaze data produced by an eye tracker can be used to evaluate how users interact with visual impressions being presented to them. Therefore, eye trackers are used in a wide variety of areas, ranging from gaming to various research areas. For example, in software engineering research, eye trackers have been used to study various tasks such as source code reading and debugging, comprehension of software artifacts, e.g., source code and UML class diagrams, and software traceability [6].

This thesis proposes a feedback system utilizing eye trackers as a tool to improve RPP. The system will mitigate missing non-verbal cues and gestures by using gaze data provided by an eye tracker. At the same time, the gaze data collected will be used to evaluate the feedback system using eye-related metrics.

Cognitive load

It is well established that working memory can only process a limited amount of information at a time [7]. Therefore, limited memory affects learning because learners must allocate their working memory to process both information which leads to learning and information unrelated to learning.

Cognitive load theory (CLT) is a framework that categorises working memory into different kinds of loads experienced during learning [3]. CLT states that *extraneous* cognitive load – load imposed by inappropriate instructional design related to the learning material – does not contribute to learning. For example, the extraneous load can be imposed by the split-attention effect, where a learner splits their attention between multiple information sources [8, 9]. Therefore, when programmers experience cognitive overload, the extraneous load must be reduced because overload hinders learning and understanding of the task at hand.

Thus, the feedback system should provide help when the programmers' cognitive load is too high. The help provided aims to reduce the extraneous load by improving the presentation of instructional information related to the pair programmers' task. Additionally, the feedback should reduce the cognitive load imposed by splitting their attention when communicating and coordinating location

in the source code.

Feedback systems

We define a *feedback system* as a Computer Supported Cooperative Work (CSCW) system for providing timely and meaningful feedback. Feedback systems can be categorized based on who receives the feedback and the type of feedback given to the users. For this thesis, we differentiate between *feedback to self* – where feedback is given to yourself – and *feedback to partner* – where feedback is given to your partner. In addition, we recognize two different types of feedback, namely behavioral feedback and instructional feedback.

Previous research has evaluated and established the benefits of shared gaze in feedback systems for remote cooperation [10–19]. However, few studies have evaluated gaze awareness in a programming context. Stein and Brennan [20] found that seeing a person's gaze solving a debugging task before solving the same task yourself (asynchronously) reduced the time spent finding the bugs. D'Angelo and Begel [10] used real-time shared gaze in an RPP context and found that gaze awareness increases JVA and improves referential communication.

For this thesis, we design a CSCW system that provides high-quality real-time feedback to remote pair programmers. The feedback system targets pair programmers who are working synchronously and remotely. The programmers cannot see their partner's screen, so the system aids the pair in communicating location through gaze sharing and encourage pairs to align their focus – called behavioral feedback. In addition, the system provides instructional feedback to ensure that the programmer's cognitive load is not overloaded.

1.2 Problem statement

This thesis aims to design, develop, and evaluate an intelligent feedback system that utilizes eye tracking in a remote pair programming (RPP) context. The goal is to develop an effective and efficient system to provide programmers with helpful feedback when facing difficulties in synchronous RPP. In addition, since eye trackers have become affordable and ubiquitous, we explore the use of low-end eye trackers to gather gaze data from pair programmers and utilize this data to provide real-time feedback.

The feedback system provides two distinct types of feedback in real-time – behavioral feedback and instructional feedback. The behavioral feedback will provide gaze awareness by visualizing partners' gazes. The instructional feedback will help

to debug software by giving the pair hints when their cognitive load exceeds a certain threshold or providing instructions to align their attention when their gazes have not overlapped during the last minute. Feedback will be provided based on three gaze measurements – gaze coordinates, joint visual attention (JVA) and cognitive load. Behavioral feedback is provided based on gaze coordinates, while instructional feedback is triggered based on their individual cognitive load and the lack of JVA. The system will be developed for Visual Studio Code (VS Code), a text editor with rich extension support.

Our study differentiates itself from previous studies on feedback systems by providing different types of feedback – when the partners' cognitive load is high or when their JVA is low. The system will help programmers with a high cognitive load by giving them instructional feedback with hints on how they can solve the bugs. The system must be intelligent, meaning that it decides automatically (in real-time) when to give feedback. We are not aware of any existing feedback system which aims at solving the aforementioned issues related to RPP and by providing instructional and behavioral feedback.

We define the following research questions:

RQ 1.

How do timely hints and feedback on cognitive load (instructional feedback) impact the performance of remote pair programmers?

H1. Remote pair programmers using the feedback system will have a significantly higher *performance* than pair programmers not using the feedback system.

H2. Remote pair programmers using the feedback system will have significantly lower *cognitive load* than pair programmers not using the feedback system.

RQ 2.

How does the knowledge of a partner's gaze attention (behavioral feedback) impact collaboration?

H1. Visualization of the partners' gazes will significantly increase the level of joint visual attention (JVA) compared to the condition without the visualization.

1.3 Structure of the thesis

Firstly, in Chapter 1, the motivation behind this thesis is presented along with its research questions. Also, it presents the current gap in research and how this

thesis contributes to closing this gap. Chapter 2 provides an overview of related work and how the relevant research was found and deemed relevant. It further motivates our thesis and proposed feedback system and how it contributes to the existing research. Chapter 3 describes and justifies this thesis's research design – how data will be collected and analysed. It also provides an overview of the implementation details of the feedback system and its features. Lastly, it informs researchers on how to recreate the feedback system. Chapter 4 presents our observations and findings from the data collection and the analysis. In Chapter 5, the implications of the results are discussed, along with limitations, further work, and the findings' generalizability. Lastly, in Chapter 6, a summary of this thesis's work and contribution is given.

Chapter 2

Background and Related Work

2.1 Paper selection

We used a combination of database searches and the backwards snowball method to conduct the background research [21]. Google Scholar[1] was used as the database for finding papers. We searched for papers containing keywords related to this thesis' objectives — *eye tracking*, *gaze*, *gaze awareness*, *gaze sharing*, *feedback* and *remote pair programming*. The title of the papers was used as the first criteria for selecting those that appeared relevant. The papers which we deemed relevant were used as the starter set for the backward snowball method; by reading the reference list of each paper, we determined which seemed relevant or not. Once we had gathered roughly 130 papers, we read through their abstracts to make sure that the papers fulfilled our criteria, which were:

- the paper must be relevant to our objectives
- the paper is peer-reviewed and published
- empirical evidence or evaluation was used to prove their findings

In the following sections, we present key findings from our literature review.

2.2 Joint Visual Attention

A critical factor in learning and collaboration is achieving a *joint focus of attention* [5, 20]. Joint attention is defined as 'the tendency for social partners to focus on a common reference and to monitor one another's attention to an outside entity, such as an object, person, or event. [...] The fact that two individuals are simultaneously focused on the same aspect of the environment at the same time

[1] https://scholar.google.com/

does not constitute joint attention. To qualify as joint attention, the social partners need to demonstrate awareness that they are attending to something in common' Tomasello [22, pp. 86–87].

In a face-to-face situation, the gaze is a vital nonverbal cue for communication, and social interactions [23, 24]. Unfortunately, working remotely can offer some difficulties in becoming aware of a partner's focus of attention. Remote work can cause problems when communicating and referencing points of attention, which is an integral part of RPP. For example, discussing elements in a source code document involves frequent referring to a specific part or line of source code. As a result, remote pair programmers must often verbally refer to source code locations. Unfortunately, verbal descriptions of an object of interest are susceptible to misinterpretation [25]. However, Zhang *et al.* [2] found that gaze can be used as a natural input source for referential communication and improves collaboration in co-located work.

Overlapping gaze between collaborating partners has been shown in prior studies to indicate the quality of interaction and comprehension [26]. According to Whittaker and O'Conaill [27, pp. 23–49], JVA, the synchronization between two individuals' gazes, is critical for effective collaboration and decision-making. Kütt *et al.* [28] also found that gaze sharing between partners enhanced partners' attention awareness and encouraged joint attention.

Our study seeks to investigate the role of JVA in remote collaborative tasks. Effective collaboration and the quality of interaction and comprehension have been proven to be tied to JVA [5, 14, 26]. Therefore, our proposed feedback system will provide remote pair programmers with gaze awareness by visualizing gaze in their text editor. The goal is to give both partners an indication of where each other is looking and an easier way of referring to locations in the source code. In addition, the system will provide feedback to the pair when their JVA falls below a certain threshold. By giving pairs instructional feedback when their attention is not aligned, we hope to see an increase in the pairs' JVA and thus increase their performance and collaboration quality.

2.3 Cognitive load

CLT is a framework built on the notion that we experience different kinds of load when learning [3]. It describes instructional design implications and how to manage working memory load to ensure successful learning and performance. Paas *et al.* [29] categorized three different dimensions of cognitive load — *intrinsic*, *extraneous* and *germane* load. *Intrinsic* load measures the load required for the task

at hand, and *germane* load is associated with cognitive activities that enhance learning. Therefore, the germane load should be increased to enhance learning [30].

Extraneous load measures the cognitive load imposed by activities that do not contribute to learning. This load is imposed by a sub-optimal format of the learning task at hand and the presentation of information related to the task. By definition, the extraneous load must be decreased because it hinders learning by unnecessarily increasing the limited cognitive capacity. Therefore, this is the most relevant dimension of cognitive load for our thesis.

We seek to decrease the *extraneous* load by providing feedback when the pair programmer's cognitive load reaches a certain threshold. Prior research has found that measurement of cognitive load allows a system to adapt to the user's cognitive state and modulate task difficulty [31, 32]. Duchowski *et al.* [32] also concluded that being able to distinguish users' cognitive load has significant design and evaluation implications for interactive systems. In addition, by presenting each partner's gaze in the text editor, we hope to reduce the cognitive load imposed by their split attention when communicating and coordinating location in the source code.

While measuring cognitive load, it is important to accurately estimate how difficult a person perceives the task at hand and how much mental effort the subject uses and do this without distracting them. Measuring cognitive load can play an essential part in providing feedback to users to avoid overloading them. We will use cognitive load as a variable for measuring how difficult or how much mental effort participants experience in solving the task at hand. Our feedback system will provide instructional feedback based on the pair's cognitive load, and we expect to see a decrease in cognitive load when this feedback is triggered.

2.4 Using eye tracking to estimate cognitive load

Eye movement, such as pupil dilation, fixations and blink rate, have been shown to detect cognitive states [33]. Task difficulty was first shown to correlate with pupil diameter by Hess and Polt [34], who found that pupil size increases with working memory load and executive load. It has since become one of the most popular indicators of cognitive load. Several studies have proved pupil size to effectively estimate the momentary load on a subject [34–38]. Kramer [38] argued that pupil dilation could provide a reliable measure for processing demand in general. Schulte-Mecklenbeck *et al.* [37] argued that fixation duration and pupil dilation could be strong indicators of emotions, stress or cognitive load. Kahneman and

Beatty [35] also showed that more difficult memory tasks induced larger pupillary response, further supported by Krejtz *et al.* [36].

Measuring pupil diameter can be a sensitive process. Several factors can affect pupil size, such as gender, anxiety, novelty, and light fluctuations [39]. Estimating cognitive load without interacting physically and thus distracting the subject during the experiment is critical for accurately estimating it. Therefore, we must take precautions to ensure that the cause of pupil responses from external factors is limited.

The use of eye trackers to track cognitive load through pupil diameter has gained traction due to their recent improvements in both affordability and accuracy. In addition, eye trackers can be used to measure pupil dilation continuously with low latency [40]. The low latency means that the eye tracker data stream reflects how pupil size reacts to visual information and the task at hand near real-time.

Additionally, inferring cognitive load based on pupillometry has the added benefits of being a non-invasive measurement with eye trackers and is less expensive than other physiological methods, e.g. EEG [40]. Also, according to Pietinen *et al.* [41], recording eye-related metrics in a PP situation must be done unobtrusively and mimic a realistic setting. The programmers should be free to move their heads or bodies as they would do in a natural setting.

When designing our feedback system, we will ensure to record eye metrics unobtrusively using eye trackers. The programmers will be free to move their heads as they wish, avoiding unnecessarily increasing users' cognitive load. The choice of an eye tracker can also impact pupil diameter measurements. Our feedback system will use the Tobii Eye Tracker 4C. Because this is an older model released in 2016, we do not get the benefits of the latest improvements in eye-tracker technology, which can give less accurate data measurements than newer eye trackers [42, 43]. For example, the Tobii Eye Tracker 5 has a higher gaze recording frequency and has improved accuracy during head movements [44].

One of the most prominent approaches for calculating cognitive load using pupil diameter has been to compare diameter relative to a baseline value [32]. However, this baseline-related measurement imposes several problems due to head placement changes and illumination changes [32]. Therefore, Duchowski *et al.* [32] propose an alternative to the baseline-related metric, which estimates cognitive load based on relative moment-to-moment pupil dilation, called Index of Pupillary Activity (IPA). IPA provide an alternative to eye-tracked baseline-related pupil measures. The metric is based on relative moment-to-moment pupil size and shows how IPA discriminates between task difficulty relative to cognitive load. The

IPA is a technique that provides a psychophysiological measurement of cognitive workload based on changes in pupil dilation. It is a wavelet-based algorithm relying on wavelet decompositions of the pupil diameter. The IPA increases significantly with task difficulty, and Duchowski *et al.* [32] suggests that the measurement of pupil oscillation is a good alternative for estimating task difficulty and cognitive load. They also provide complete documentation of the IPA algorithm and its calculation.

Because of the IPA's accuracy in measuring cognitive load and it being well-documented, we chose the IPA measurement to calculate cognitive load in our feedback system. However, as Duchowski *et al.* [32]'s research focused on evaluating the effectiveness of IPA and how well it measured cognitive load, they performed the calculations post-experiment. Therefore, in our contribution, we propose a system that calculates cognitive load continuously in real-time to provide cognitive-load-based feedback to the system's users.

2.5 Feedback-tools using eye tracking for collaborative tasks

Using eye-tracking data to provide feedback to programmers collaborating on specific tasks has been proven to increase the quality of collaboration, and it significantly improves the effectiveness of communicating locations in the source code [10, 45]. Providing users with feedback to indicate a partner's gaze location has been suggested as an effective method to improve both communication and JVA [10]. In the experiment performed by D'Angelo and Begel [10], participants expressed in post interviews that discussing locations in the code was perceived as more effective with the feedback tool enabled.

Stein and Brennan [20] tested the hypothesis that one person's visual focus of attention represented as a cursor could be helpful in a software-debugging setting. The experiment consisted of two phases. In the first phase, the participants, consisting of professional programmers, searched for bugs in small Java programs. In the second phase, a new group of participants were able to see the gaze of the first group, who had solved half of the bugs in the source code. The second set of programmers found the bugs more effectively when they could see the gaze of the first group, suggesting that another person's gaze can be a valuable indicator for another person in a software debugging task. The gaze visualizations may have helped by pointing out the bugs' locations in the source code and limiting the search area. Our study will apply their findings and take them one step further by investigating how synchronous gaze awareness affects collaborative interaction

in an RPP context.

Gupta *et al.* [15] conducted a user study to test a system prototype for remote collaboration. They explored the effect of sharing pointing cues and gaze using a head-mounted camera, eye-tracking camera and a head-mounted display. In the experiment, participants collaborated as pairs. One of the pair's partners was categorized as the local worker and the other as a remote helper. The task involved constructing three-dimensional structures using LEGO Duplo pieces. They used four conditions in the experiment. Only live video and audio were shared in the *no cue* condition, while the *pointer cue* condition gave feedback as a green virtual pointer. The *eye-tracker cue* displayed the workers' gaze on the helper's monitor to show their focus of attention, and *both pointer and eye-tracking cues* condition enabled the cues from all three conditions. Their qualitative feedback indicated that the pointer and eye-tracking cues in conjunction helped the participants perform the task significantly faster. Furthermore, the *eye-tracking cue* condition saw the same result as the *pointer cue* condition in significantly improved quality of communication, collaboration and co-presence. Additionally, they found improvements in the worker's focus and the helper's enjoyment.

The use of eye-tracking data in feedback systems when collaboratively solving tasks has yielded undoubtedly beneficial results [10, 15, 20]. We need to consider their design methods and conclusions when designing our new system. These are their key takeaways: D'Angelo and Begel [10] found that communicating code locations was more effective with a feedback system enabled. Stein and Brennan [20] suggested that providing feedback could minimize the search area and make more efficient bug locating. Gupta *et al.* [15] saw significantly faster task completion time with eye-tracking based feedback. These findings further motivate our study, and we expect to see similar results with our proposed feedback system. We take Stein and Brennan [20]'s study one step further by incorporating real-time feedback from the partners' gazes.

2.6 Visualizing gaze

According to Müller *et al.* [12], gaze visualization must be designed based on the task characteristics. Several studies evaluated gaze visualisations [2, 10, 12, 18, 46, 47].

Maurer *et al.* [18] investigated the use of shared gaze in an online cooperative game. They found that the gaze can be used for communication in games, but the participants had an issue with continuously visualizing the gaze. Continuous visualization confused the cooperating participants since they could not determine

when the gaze was used as a means for communication or just unwanted input. Therefore, they suggest that the players can toggle the visualization on and off to mitigate this issue. We must carefully consider when and how to visualize gaze to avoid making it distracting for our study's purpose. Therefore, the feedback system should give the user control over toggling gaze sharing on and off.

D'Angelo and Gergle [46] evaluated the effect of various gaze visualizations on communication processes and their influence on collaborative performance. In a collaborative searching experiment, participants tested different gaze visualization for visualizing their partner's gaze. The *shared area* visualization showed when the partners' gazes were overlapping as a black outlined circle. In contrast, the *heat map* visualization depicted where participants had previously looked. Lastly, the *gaze path* visualization depicted a trail of where the participants had looked in the last three seconds.

The path and heat map visualizations were always enabled, while the shared area visualization was only visible when the partners' gazes overlapped. They found that constantly showing the path visualization was useful, although distracting. This visualization supported the quick coordination of object locations. The shared area visualization improved coordination and perceived utility without distraction, as it only activates when their gazes overlap.

They suggest that partially available gaze visualizations should be further explored because the style is new. Therefore, we will also implement a shared area visualization but not the gaze path visualization as it is too distracting. The heat map visualization was the least useful and the most distracting – participants were slower when solving the tasks with the heat map. Therefore, we will not visualize gaze as a heat map in the feedback system. Although, compared to the no visualization control group, some improvements in communicating object locations were found.

Zhang *et al.* [2] investigated gaze sharing for co-located collaboration between partners in a searching task using the same screen. They evaluated different gaze indicators and how they affect coordination and communication between the partners. The results showed that even though gaze enhances collaboration in this specific task, the visualizations distracted the participants. There was a trade-off between visibility and distractions, where task performance was influenced by how the gaze was visualized. The gaze path visualization was the most distracting and challenging to interpret for the participants. They also gave the participants control over toggling the gaze on and off. While 15 out of 20 participants left the gaze indicators on without ever turning them off, others toggled them off when their gaze was thought to become too distracting, to signalize where the partners

should look or to hide from the partner where they were looking. Even though this study evaluated gaze visualization in a different context than ours, they highlight the importance of preventing the visualization from being too distracting. Therefore, users of our system will have control over when to display gaze.

Visualizing the gaze in a document's left or right margin is an excellent way to visualize the gaze in a remote programming context [10, 47]. Hill *et al.* [48] investigated scroll bar visualization of *computational wear* of a document. They argue that the relative mapping of the document height in the scroll bar is suitable for providing an overview of placements and locations in the document. Additionally, it collocates positional information with navigation controls, meaning users can easily navigate to the location to which the information is related.

Yao *et al.* [47] visualized students' gazes in a programming course to support remote learning. One of the evaluated visualizations was the gaze indicated in the scroll bar. The scroll bar visualization was helpful since the code spanned over the screen's height. In addition, because students could be working on different parts of the code, this visualization made it easier for the teacher to get an overview of every student's gaze.

D'Angelo and Begel [10] visualized gaze in a remote pair programming experiment where the participants were given source code refactoring tasks. The visualization showed the current location of their partner's gaze in the code. When the partners looked at the same location in the code, the color of the visualization changed. The visualizations were directly shown in the code editor in the left margin as bars. Their visualization increased JVA and used more implicit references than explicit ones in their communication while being much faster at responding to the references and communicating more efficiently. In a collaborative problem-solving task, Schneider and Pea [49] also found that gaze sharing increases JVA and that JVA is positively correlated with learning gain.

Research has shown that constant sharing of real-time gaze is helpful, although distracting with the current form of visualizations that have been tested [11, 46].

The visualization must show where each pair programmer is working and indicate whenever their gazes are aligned. These visualizations must be unobtrusively shown so that there is no ambiguity or uncertainty when using gaze for spatial referencing. Based on D'Angelo and Begel [10] and Yao *et al.* [47] results, our feedback system will visualize the partner's gaze in the left margin of the document and in the scroll bar in what we believe to be a non-distracting way. The visualization of gaze in the scroll bar is proportionate to the length of the document, and its position will vary based on the number of lines of code in the

document. Having a visualization in the left margin could make it more transparent where their partner is looking precisely in the code document and when they are looking at the same area of the code.

A summary of the gaze visualization findings, and their implications on its design for our study, is presented in Table 2.1.

Table 2.1: Gaze visualization design decisions

Findings	Studies	Design decision
Continuously visualizing gaze is distracting and causes confusion.	[2, 18]	User decides when to toggle gaze on or off.
Visualizing gaze overlap is useful without being distracting.	[10, 46]	The gaze visualization will indicate whenever partners' gazes overlaps.
Visualizing gaze in the document margin is a good and useful visualization in a remote programming context.	[10]	Gaze will be visualized in the document margin (next to the line numbers).
Visualizing gaze in the scroll bar is useful for seeing the location in a document which is longer than the height of the screen.	[47, 48]	Gaze will be visualized in the scroll bar.

2.7 Signaling principle

As referred to by Van Gog [50], the signaling principle is the effect of multimedia learning material becoming more effective when cues or signals are added to guide learners' attention to the relevant elements or sections of the material being presented. Mayer [51] conducted a review of two studies by De Koning *et al.* [52] and Boucheix and Lowe [53], which explored different aspects of the signaling principle with the help of eye-tracking data.

Boucheix and Lowe [53] studied how signaling cues affected learners while playing piano and De Koning *et al.* [52] performed a study on visual attention

around animations of the cardiovascular system. Eye movements were measured in the presence and absence of visual cues. Both studies found that learners spent more time looking at important features when highlighted. The studies also found a strong link between eye-fixation time and learning outcomes. Mayer identified 'spreading color cues', which highlighted the relevant features of an animation as the visual signaling that improved learning outcome or cognitive processing.

Kiili and Ketamo [54] performed a study on a game-based learning process. Based on perceptual data, they evaluated the effectiveness of cognitive feedback. The result indicated that the sooner players noticed the cognitive feedback (signaling cue), the better they understood the game and the meaning of the feedback. Kiili and Ketamo's results further support Van Gog [50]'s conclusion on the appropriate and strong highlighting of vital elements.

In our proposed feedback system, users will receive a signaling cue when a pair's JVA falls below a certain threshold and continuously visualize the partner's gaze in the editor, which can also be viewed as a signaling cue. Boucheix and Lowe [53] and De Koning *et al.* [52] found that highlighting increases time spent looking at important features. In addition, Kiili and Ketamo [54] and Van Gog [50] further supported appropriate and strong highlighting of necessary elements. Therefore, with the proper use of the *signaling principle*, we can guide the pairs' attention by highlighting their partner's gaze location to make it clearer that they are focusing on the same area of the code.

2.8 Remote pair programming

Pair programming is a collaborative method where two programmers do programming activities on the same code and display. Examples of such activities are design, review and debugging. This method of collaboration is said to improve productivity, and the quality of software products [55]. However, it requires coordination and a shared understanding between the programmers.

Pair programming is usually done with two co-located programmers while they share a display. However, Baheti *et al.* [56] showed that the co-located factor could be neglected. They reviewed prior literature where the same benefits had been reported when pairs worked spatially distributed. Pair programming is a method which is well suited for studies of shared gaze awareness as the task is synchronous and collaborative [10]. It is also a tightly coupled collaborative task where physical affordances such as hand gestures and non-verbal cues play an essential part in the collaborative process. Therefore, it is important to consider how gaze is visualized, as attending to unintentional visualization of eye movement is

disruptive [11, 46].

Remote work has seen a steady increase in popularity over the last decades, especially during the last three years, with the pandemic converting large parts of the workforce to work from home [1]. Technological advances have improved the sustainability and efficiency of remote work. However, some of the physical affordances we experience with co-located work, such as non-verbal cues, are lost in a remote work environment [10]. Working remotely in groups or pairs affects the ability to understand where one's partner is looking. However, dual eye-tracking solutions could help mitigate the lack of these abilities by sharing gaze information between partners and thus incorporating non-verbal cues into remote work [57].

RPP is a commonly used practice where developers work in pairs on a single project where they do not sit next to each other. This method has been proven in previous work to provide higher quality software, team cohesion, faster deliveries and a high perceived value by the programmers [58, 59]. A survey conducted by D'Angelo and Gergle [46] found that developers would like the flexibility of RPP. However, the lack of existing tools failed to create an environment where it could be done efficiently. Our work applies the dual eye-tracking methodology to RPP and presents a feedback system to extend RPP's current state. RPP is also a method which is well suited for studies of shared gaze awareness as the task is synchronous and collaborative. Our thesis evaluates our tool for sharing gaze and providing feedback for pair programmers in a simulated remote setting.

D'Angelo and Begel [10] studied *remote pair programming (RPP)*, where their visualization of gaze was based on a mirrored display. Our solution supports each programmer having their own separate view of the system, enabling pairs to collaborate on a source code base without mirrored displays or screen sharing.

Chapter 3

Methodology

3.1 Research design

First, we design and create a feedback system based on previous research findings and implement a novel cognitive-load-based feedback mechanism. Then, we conduct a true experiment to identify the potential effects of the feedback system on pair programming. The feedback system will be the experiment's artefact. After that, we perform a repeated-measures experiment using a within-subject research design and collect quantitative and qualitative data. Finally, we analyse the data collected.

3.1.1 Research strategy

This study's research strategy was structured into two parts – design and creation, and experiment. For the first half of the thesis, we designed and created a feedback system that would be used as an artefact in an experiment. The experiment aimed to discover and evaluate the feedback system's effect on the users, or more concretely, to answer our research questions. According to Oates [60, p. 109], a research project that follows the design and creation strategy must show academic qualities, such as analysis and critical evaluation, to be considered research. Furthermore, it must also contribute to knowledge. Therefore, we combined the design and creation strategy with the experiment strategy to analyse and evaluate real-world usage of the feedback system, thus contributing to knowledge. However, the design and creation strategy can also contribute to knowledge on its own, by exploring and implementing technical novel ideas. Therefore, we aimed to contribute to knowledge by implementing and evaluating a new type of feedback not previously researched. The new type of feedback is explained in Figure 3.15, namely cognitive-load-based feedback.

3.1.2 Design and creation

The goal of the feedback system is to help remote pair programmers debug by improving the communication between them and reducing their cognitive load. The feedback is based on gaze data gathered using a dual eye-tracking setup with Tobii Pro 4C trackers[1]. Each pair programmer will have an eye tracker connected to their computer. The system is intelligent, meaning that the data gathered is analyzed in real-time to provide automatic feedback while the programmers are debugging. The feedback is provided directly in the text editor they are using.

VS Code[2] is built with a focus on extensibility; its extension application programming interface (API)[3] makes it ideal for implementing a feedback system imposed in the text editor. Therefore, the feedback system is implemented using the VS Code text editor. To provide remote collaborative editing and debugging capabilities for the collaborators, the system uses Duckly[4]. Duckly is a VS Code extension which enables real-time sharing of code between pair-programmers synchronously collaborating remotely. The extension visualizes each programmer working in the editor with a coloured cursor of where the users cursor is currently located. Marked text is also highlighted to the partner. We also considered using Live Share, a VS Code extension created by Microsoft, allowing programmers to share and collaborate on projects while debugging jointly remotely. However, one limitation of using Live Share is that it indicates each user's location in the overview ruler (scroll bar), which our system also does. Therefore, Live Share conflicts with our feedback system's visualization. Thus, we opted for Duckly instead.

To work with and access the data from the eye trackers, the Tobii Pro SDK is used. The SDK needs to pass data to the VS Code extension. Therefore, because the SDK is written in Python, data is shared between the extension and the SDK through websockets[5].

We designed the feedback system and gaze visualization based on the literature review findings described in Chapter 2 and outlined in Table 2.1. The system provides real-time gaze awareness to the pair collaborating in the overview ruler (scroll bar) shown in Figure 3.4 and in the document border, next to the line numbers shown in Figure 3.1. However, the visualization in the document border is only visible to the partners if they have an overlapping viewport. Therefore, they will not know where each other is looking if they are not scrolled to the same area.

[1] https://web.archive.org/web/20191219134435/https://gaming.tobii.com/tobii-eye-tracker-4c/

[2] https://code.visualstudio.com/

[3] https://code.visualstudio.com/api

[4] `https://marketplace.visualstudio.com/items?itemName=gitduck.code-streaming`

[5] `https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API`

We also visualize gaze in the overview ruler to counteract this missing information since the overview ruler height is relative to the document height. Thus, the gaze location will always be visible to the partner.

The document border visualization is nine lines high to allow a margin of error. Thus, we add four lines on either side of the gaze coordinate from the eye tracker. Because each line in the document is 19 pixels high, the visualization is $19\text{px} \cdot 9 = 171\text{px}$ high. Once the participants' gaze overlaps, the visualization changes color to green, illustrated in Figure 3.1a, where the visualization is pastel blue and green in Figure 3.1b. We carefully designed the gaze visualization to avoid becoming more distracting than informative. The colors were changed from bright red and green to a more muted pastel blue and green based our initial pilot testing feedback. As we wanted to reinforce the importance of JVA, the visualization color changes to green.

We define two non-functional requirements for Eye Tracking Studio (ETS): performance and usability. The performance of the feedback system must be high to handle real-time remote collaboration with a shared gaze. Usability is important for the users to avoid the system being difficult to use or more distracting than helpful. These requirements are summarised in Table 3.2.

The eye tracker generates gaze data with a 90 Hz refresh rate. However, rendering the gaze 90 times each second makes the visualization appear very bouncy and choppy. Therefore, to make the gaze sharing appear more smooth, we implemented an exponential moving average calculation of which lines to draw the gaze in the text editor document. In addition, we limited the rendering rate to 60 Hz, which reduced the load on the system, to fulfill NFR01. Another measure to reduce distraction was the toggling gaze visualization on or off feature, using the button in the activity bar, as shown in Figure 3.3. This feature allows users to turn off the visualization whenever they feel that it becomes too distracting, fulfilling NFR02.

When clicked, the button *Jump to partner* moves the editor view to the partner's gaze location in the document. This feature was also accessible with the shortcut (*ALT+P*). Alternatively, they could click on the gaze visualization in the overview ruler. As Figure 3.2 shows, the instructional and behavioral feedback is presented as a modal covering the editor and forces the user to respond. We decided to provide feedback as a modal due to the VS Code API guidelines stating that a modal should be used when immediate user input is needed and only one input is needed[6]. We need immediate user input to give them hints for the debugging tasks. In addition, during the pilot trials of the feedback system, the

[6] `https://code.visualstudio.com/api/references/extension-guidelines#notifications`

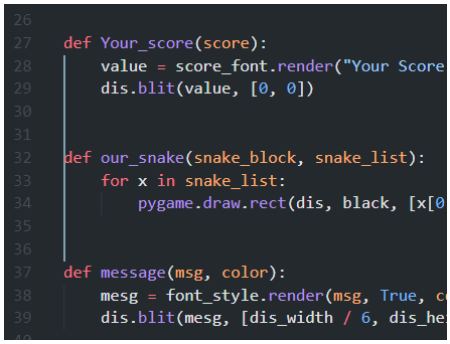
participants did not always notice the notifications when they were shown as a toast notification in the lower right corner, which argues for a modal alert. The system gives the users feedback on what they should do to improve their collaboration. The feedback is instructions to align their focus of attention. In addition, when the system noticed their cognitive load was high during the experiment, we physically gave them hints for finding and solving the bugs in the source code. A summary of the system's functional requirements is given in Table 3.1.

Table 3.1: Features of the feedback system (functional requirements). Does not include features provided by third party applications (such as shared document and cursor).

ID	Feature	Description
FR01	Host session	Start a gaze sharing session.
FR02	Join session	Join a gaze sharing session.
FR03	Stop session	Stop and leave a gaze sharing session.
FR04	Gaze document margin	Visualize partner's gaze in next to the document lines.
FR05	Gaze in overview ruler	Visualize partner's gaze in the overview ruler (scrollbar).
FR06	Joint visual attention visualization	Indicate that the partners have achieved joint visual attention.
FR07	Joint-visual-attention-based alert	Give feedback to the partners that they need to align their gaze.
FR08	Cognitive-load-based alert	Give feedback to the partners that they can receive hint to reduce their cognitive load.
FR09	Jump to partner	Scroll to partner's gaze location in document.
FR10	Toggle gaze	Turn the gaze visualization on or off.

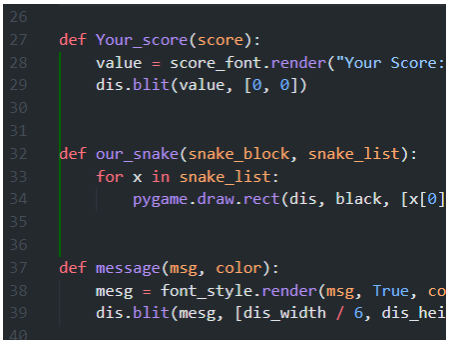
Table 3.2: Attributes of the feedback system (non functional requirements).

ID	Requirement	Description
NFR01	Performance	Sharing gaze should happen in real time (avoid lag and delays).
NFR02	Usability	Should not be invasive or intrusive.



```
def Your_score(score):
    value = score_font.render("Your Score
    dis.blit(value, [0, 0])


def our_snake(snake_block, snake_list):
    for x in snake_list:
        pygame.draw.rect(dis, black, [x[0

def message(msg, color):
    mesg = font_style.render(msg, True, c
    dis.blit(mesg, [dis_width / 6, dis_he
```

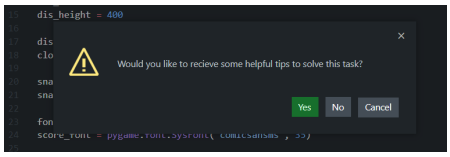
(a) Border visualization of gaze.



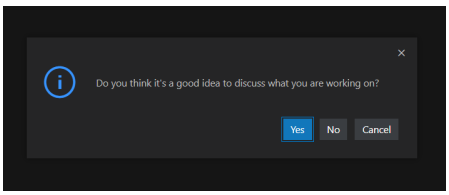
```
def Your_score(score):
    value = score_font.render("Your Score:
    dis.blit(value, [0, 0])


def our_snake(snake_block, snake_list):
    for x in snake_list:
        pygame.draw.rect(dis, black, [x[0]

def message(msg, color):
    mesg = font_style.render(msg, True, co
    dis.blit(mesg, [dis_width / 6, dis_hei
```

(b) Border visualization with joint visual attention.

Figure 3.1: Gaze visualized in the left margin of the text document – with and without joint visual attention.



(a) Cognitive-load-based alert.



(b) Joint-visual-attention-based alert.

Figure 3.2: Joint visual attention and cognitive load alert notification styles.

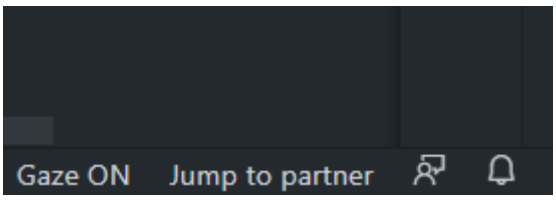
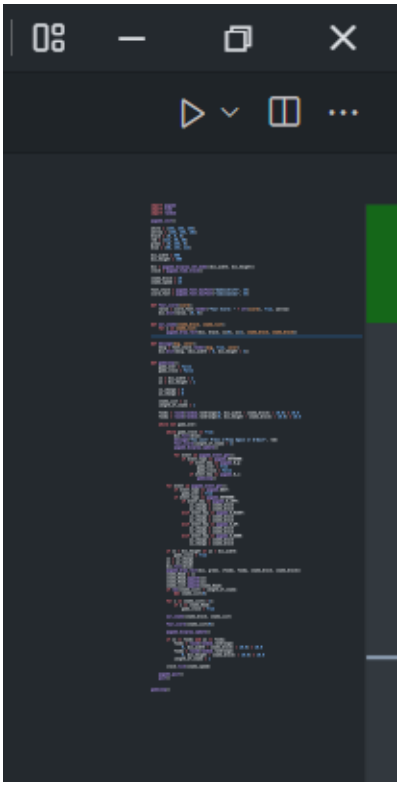


Figure 3.3: Activity bar showing the *Jump to partner* and gaze toggle buttons.

(a) Overview ruler (scrollbar) visualization of gaze.



(b) Overview ruler (scrollbar) visualization of gaze with joint visual attention.

Figure 3.4: Gaze visualized in the left margin of the text document – with and without joint visual attention.

3.1.3 Experiment

In the second half of the research strategy, we conducted an experiment using the ETS feedback system in a simulated remote pair programming (RPP) context.

Participants

We recruited participants to test our feedback system. Our target group was students with a basic understanding of programming and debugging in the Python programming language. First, we recruited participants using self-selection sampling [60]. We posted the experiment sign-up form on various physical and online bulletins at the Norwegian University of Science and Technology (NTNU). However, the sign-up rate was meager, so we invited participants using convenience sampling, which yielded a higher sign-up rate.

Forty participants were recruited, yielding 20 pairs. Sixteen of the participants were students enrolled at NTNU with experience in programming and Python. The students ranged from bachelor level studies to doctoral level. Four of the participants were not students but had worked in the programming industry for three years. Every participant received a 200 NOK gift card as compensation for participating. In addition, two randomly selected participants received 500 NOK gift cards. Each pair spent one hour and thirty minutes completing the experiment. The participants were randomly assigned to pairs, or they could sign up for the experiment as a pair.

Setting and procedure

Before conducting the experiments, we performed two pilot tests to discover any major flaws with our feedback system. We discovered several things that needed improvement. Initially, we designed the gaze visualization using bright red and green colors. However, the pilot testers gave us feedback that these colors were too dominating and caused many distractions. Therefore, we changed the colors to more muted pastel blue and green. We also found a discrepancy in program complexity between the two debugging task's source codes. Therefore, we found that each bug fixed in the more complex program would be worth 1.5 points, while the other program would be worth 1 point for each bug fixed. Lastly, we found that the pilot testers did not notice when an alert was shown as a toast notification in the lower right corner of VS Code. Therefore, we changed the alert to cover the screen because we wanted to force an answer from the user.

We also used the pilot tests to verify that the instructional feedback thresholds were appropriately set. We found that 1 minute without any overlapping gaze was

suitable for providing JVA-based feedback (FR07). Additionally, an cognitive load increase of 18% in over 20 seconds was fitting for giving cognitive-load-based feedback (FR08).

During the experiment, participants were situated in the same room. However, their monitors were placed back-to-back, and a wooden board separated the pair to simulate an RPP session. Therefore, they could not see each other, but they could speak to each other freely. They were free to move their head as they wished, as in a real-world RPP context. The pairs edited the source code in a shared document and could see each other's code changes in real-time.

Each pair was exposed to all conditions in a within-design experiment structure, where the conditions are our independent variables. The two conditions are the no-feedback condition (NFC) and the feedback condition (FC), which are summarised in Table 3.3. The NFC was the control condition where the participants pair programmed without the help of our feedback system, i.e., no gaze awareness nor cognitive load based feedback. The NFC served as a baseline for comparing the feedback's effects on the pairs. For the FC, the participants pair programmed with our feedback system enabled. The condition order was counterbalanced to avoid any potential ordering effect. Half of the pairs were exposed to the feedback system enabled condition first, while the other half had the feedback system disabled in the first condition.

Every experiment session began with us introducing ourselves to the participants. Then, we gave them a consent form to sign to be eligible to participate in the experiment. The consent form informed the participants about the data, especially personal data, which we would collect from them. The experiment was structured into three parts. In the first part, the participants answered two self-report questionnaires. In the second part, they performed the two conditions, and in the third and final part, they attended an interview. The first questionnaire measured their attitude toward computers using the Computer Attitude Questionnaire (CAQ)[7], which estimates an *enjoyment* and *motivation* score for each participant. The second questionnaire measured their program comprehension and expertise with the Python programming language and its syntax. We measured their Python expertise to compare it with their task performance during the experiment. The pair spent 5 minutes answering the CAQ questionnaire and 15 minutes on the Python expertise questionnaire.

After completing the pre-tests, we calibrated the eye trackers against each participant using the 6 point calibration method provided by Tobii Eye Tracker

[7] https://iittl.unt.edu/content/computer-attitude-questionnaire-caq

Manager[8]. Then, we gave a tutorial on how the feedback system worked and explained all of the system's features. Section 3.1.2 explains every feature of the feedback system.

The text editor used the default font (Consolas, 14 pixels) and the default line-height (calculated based on font size and equals 19 pixels). The participants used two different screens, one 1920x1080 pixels screen while the other screen was 1920x1200 pixels. The feedback system accounts for different screen sizes, however they will not affect the experiment results because gaze coordinates are transformed to the corresponding line in the document opened in the text editor.

The participants debugged two Python programs while we monitored their gaze in both conditions. However, the feedback system was enabled for only one of the tasks. We handled the hosting (FR01) and joining (FR02) a session in the feedback system so the pairs could focus on debugging rather than the set-up. The pair debugged in the same document and could see each other's source code edits in real-time. Whenever the feedback system noticed the participant's cognitive load was high, we physically gave them hints for finding and solving the bugs in the source code if they wanted. They could also ask for a hint whenever they were stuck on a bug (in both experimental conditions). We gave them hints on a paper note with a source code snippet of the bug. The line that needed to be fixed to solve the bug was marked on the paper, briefly explaining its functionality. Each bug had a corresponding hint and was constant for every experiment.

They were given 20 minutes to solve the program bugs for each condition. The participants had to debug two games implemented in Python. We made our versions of the games Snake[9] and Tetris[10] as these games are so popular that our participants most likely would have played them before or be familiar with them. Being familiar with the games makes it easier for the participants to only focus on the program comprehension without also learning the game's rules and the gameplay. The bugs are only tied to the game logic, not the programming language's syntax. In addition, these games are simple to make and do not require many lines of code, meaning that the code fits into one file for each game. However, the drawback of using two different programs is that the programs might differ in complexity concerning comprehension and bug difficulty. Therefore, the programs must be similar in difficulty so that the results from each condition are comparable. Achieving similar difficulty is not easy as these are two different games with different game logic and complexity. The Tetris program had six bugs, while the Snake program had five bugs.

[8] `https://www.tobiipro.com/product-listing/eye-tracker-manager/`

[9] `https://en.wikipedia.org/wiki/Snake_(video_game_genre)`

[10] `https://en.wikipedia.org/wiki/Tetris`

Before each debugging task, we showed the participants the bug-free version of the program they were about to debug. This was to make them understand how the finished game works and how it is played. We structured the experiment so that the participants would alternate between debugging with feedback or without feedback first. However, the game they debugged with and without feedback was always constant. Tetris was debugged in the FC, while Snake was debugged in the NFC. We decided to have fixed games for the conditions to have a common baseline for comparing the test results. The pair's performance was given based on their debugging score, i.e. how many bugs they were able to fix for each condition and how much time they used.

Once the pair had completed both conditions, we interviewed them about their experience with the system. The interview lasted approximately 10 minutes and was held as a group interview.

Data collection

We gathered both quantitative and qualitative data from the experiments. We chose a mix of quantitative and qualitative data collection approaches as data regarding the system's user experience is not easily captured through quantitative methods, e.g., system log data. In addition, because experiences and feelings are not easily captured through questionnaires, we interviewed the participants to capture their emotions [60, p. 187].

Before the debugging phase of the experiment, participants answered two paper-based self-report questionnaires. They were the Python experience pretest, yielding a Python experience score for each participant and the computer attitude questionnaire for estimating motivation and enjoyment scores with computers.

The system logged usage data from the participants during the experiments. The data collected was pupil diameter from both eyes, gaze as coordinates, gaze in the program file as line numbers, IPA, timestamps of when instructional feedback was triggered and the type of instructional feedback. We also recorded audio during the task solving phase and the user's screens with a screen capturing program. Once they had completed their tasks, we saved the program files to note how many bugs they solved and analyze their debugging performance. We also noted which hints they received during the experiment. Their time on task, e.i., time spent solving the bugs, was also recorded. Table 3.4 summarises the dependent measures we collected for both experimental conditions.

User interaction with the system was also logged. Whenever the user used any of the feedback system's features, i.e., the *jump to partner* command, the *toggle gaze on and off* command and the responses to the instructional feedback, the

system logged the interaction.

After each pair had completed their debugging tasks, we interviewed them in a semi-structured group interview. We opted for a semi-structured interview because we could ask them additional questions if something came up during the interview that we had not thought about asking beforehand. Therefore, we could get a deeper understanding of the participants' experiences. The goals of the interview are summarized in Table 3.5. We used 10 minutes to interview the participants and recorded their answers with a voice recorder. The interview recording was later transcribed.

Table 3.3: Experiment conditions and independent variables.

Condition	Description
No-feedback (NF)	Feedback system disabled
Feedback (F)	Feedback system enabled

Table 3.4: Dependant variables measured during the experiment.

Data Source	Measure	Description
Gaze coordinates	Joint visual attention	Overlapping gaze between the collaborating partners.
Time on task	Task completion time	Used to measure performance of the participants.
Source code	Bugs fixed	Used to measure performance of the participants.
Pupil diameter	Cognitive load	The cognitive load imposed on the participants.
Hints	Performance	Which hints was given to each pair.

3.2 Implementation

This section describes the system implementation following the 4+1 architectural view model [61]. The diagrams are abstractions of the real system, and some simplifications of the system details have been made to improve the readability.

Table 3.5: Goals data points to collect during the interview.

Data Point	Description
User experience	Overall experience with using the feedback system.
Perceived collaboration	How users perceived the use of the system regarding collaboration.
Adoption	If the users would like to adopt the system in their day-to-day programming.
Graphical design	How the users perceived the gaze visualization.
The best and worst feature	Which features they liked the most and which they did not like and why.

3.2.1 Logical View

As shown in Figure 3.5, the logical view diagram shows the relationship between the classes of the feedback system. Only the most vital classes and relationships are modeled to improve the readability of the diagrams. In addition, every class in the frontend has access to the interfaces defined in 3.6. The interfaces have been omitted from the main diagram to improve readability. In addition, the VS Code API is also available for every class in the frontend, but this low-level API is not modeled to increase readability. Figure 3.7 shows a more detailed system diagram with the most interesting class methods and variables.

The system is structured into a frontend and a backend part, represented by the colored squares in the diagrams. We define the backend as the part of the system which reads and parses data from the eye tracker. Additionally, it deserializes the gaze data and sends it to the frontend. The backend is a Python script executed by the frontend, e.g., when a RPP session is started. The *Connection_Manager* opens a connection to the frontend WebSocket and initializes the *Gaze_Data_Producer* class, which subscribes to the eye tracker data using the low-level package Tobii Pro SDK. The *Gaze_Data_Producer* then calculates the user's cognitive load using the IPA algorithm. It also determines if the cognitive load has reached a state where the system should give feedback to the user. The gaze data, which mainly contains gaze coordinates and if an alert should be shown or not, is sent to the frontend through the full-duplex WebSocket connection. The *MessageManager* class handles the gaze data in the frontend and manages both the backend WebSocket connection and the partner WebSocket. This class serializes the incoming data and determines what to do with it. The class depends on the *GazeDataManager* class, which will use the gaze data to translate the gaze co-

ordinates to the corresponding code line in the program file, which is open in the editor. Then, the *MessageManger* deserializes the data and sends it to the partner.

The session host publishes their WebSocket Internet Protocol address (IP) to Firestore[11], a cloud-hosted NoSQL database, to establish a connection to a partner. The *FirestoreManager* class handles this. The partner can then join the session by fetching the IP and connecting to the WebSocket. Once a connection has been established, the pair will exchange gaze data. The *MessageManager* will convert the data into a *GazeData* object when a partner receives data. It also checks if any alerts should be shown, handled by *Alerts*. Finally,*MessageManager* passes the *GazeData* object to the *GazeDataManager*, which uses this object to visualize the gaze in VS Code. It also checks if the partners' gazes overlap, thus having achieved JVA.

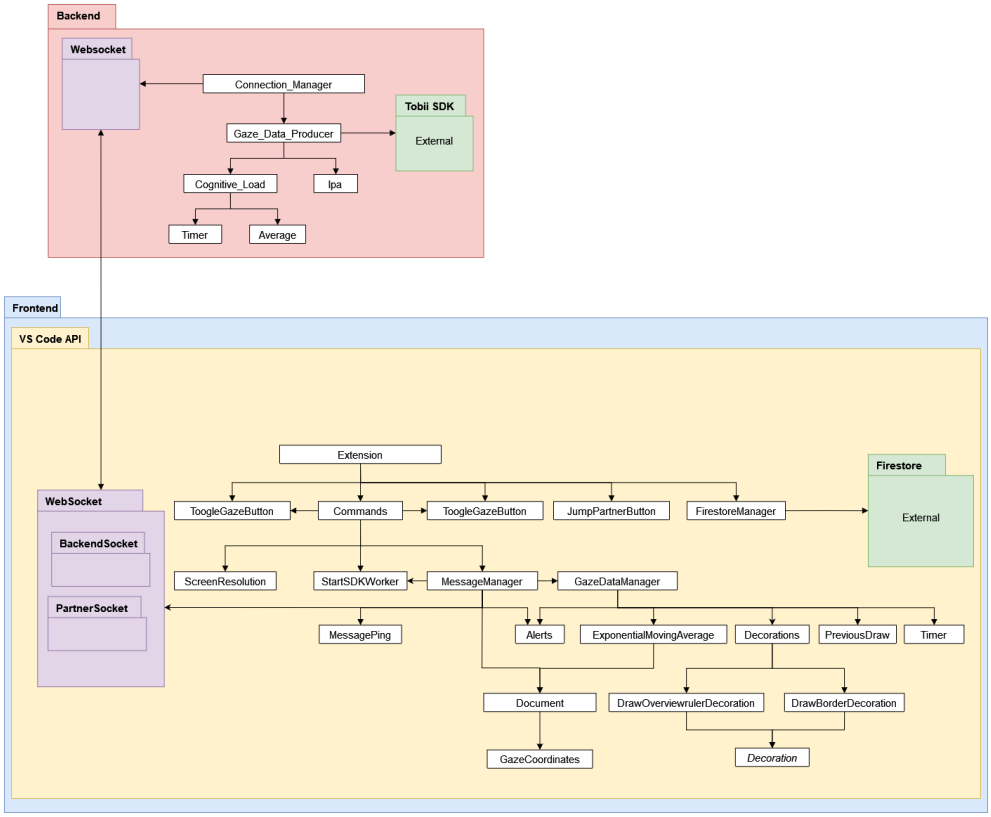


Figure 3.5: Simplified logical view (class diagram) of the feedback system.

[11] `https://firebase.google.com/`

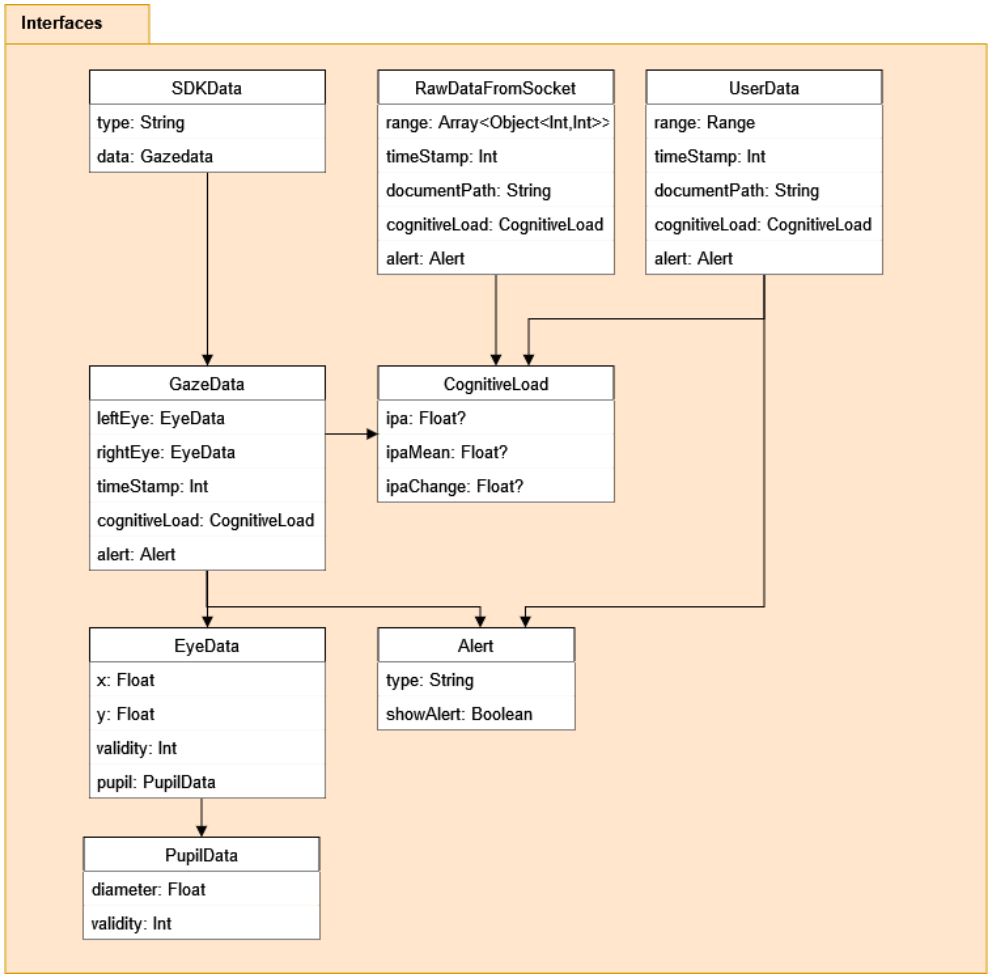


Figure 3.6: Logical view (class diagram) of the feedback system.

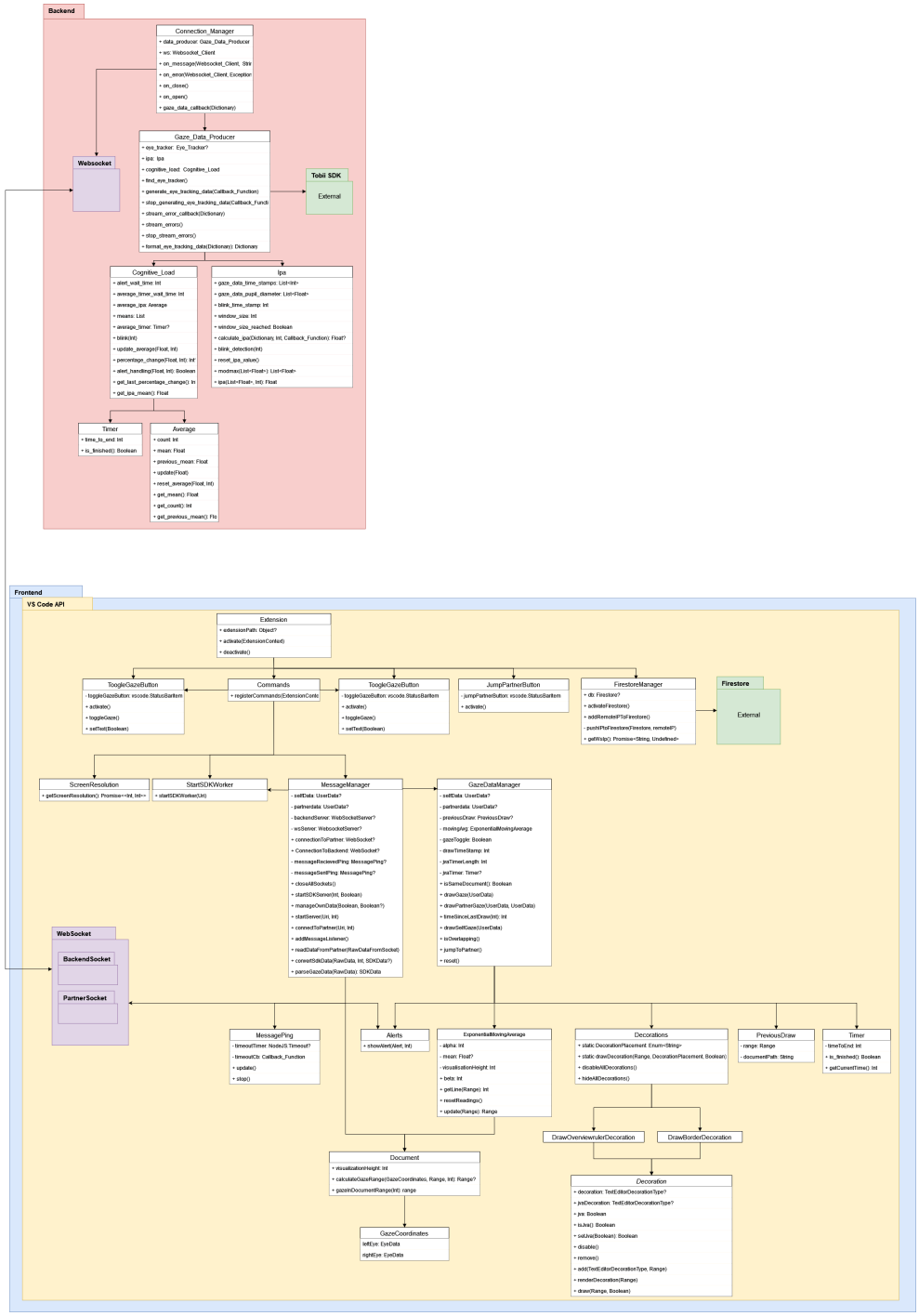


Figure 3.7: Detailed logical view (class diagram) of the feedback system.

3.2.2 Process View

Figures 3.8 to 3.10 shows the process view. The views describe the system's flow in a high-level manner where essential processes are modeled. To increase the readability of the diagrams, we split the processes over three diagrams, even though they are consecutive processes. The vertical lines in the diagrams indicate where a process is split; the processes that end in a vertical line are continued in the following figure, which starts with a vertical line. The primary process starts when a user hosts an RPP session or joins a session. Once a connection between the partners has been established, the succeeding process is equal for both partners. A more detailed description of the processes is depicted in Figures 3.11 to 3.14.

Figure 3.8 shows the hosts starting a WebSocket server, pushing the server's IP to Firestore[12] and then waiting for the partner to connect to the session. The partner fetches the server IP and then connects to the server. Once a connection between the partners has been established, they start a second WebSocket for listening to gaze data produced by the backend. The backend processes the data generated by the eye tracker, which involves calculating the cognitive load and sending the data in a format that the frontend can understand. The frontend sends gaze data to the partner once it receives it from the backend. Then, the partner visualizes the gaze data in VS Code and shows an alert if the partner had a high cognitive load or if their gazes have not overlapped during the last minute. When either partner wishes to leave the session, they execute the *Stop Session* command, and all WebSocket connections are closed, and the session is stopped.

Joining and hosting a session

Figure 3.11 shows a detailed model of the internal calls and objects of hosting and ending a session. A user starts a session by executing the *Host Session* command from VS Code. The command will use the *MessageManager* object to start a WebSocket server, which pushes the WebSocket IP to the Firestore database. Then, the user's partner can join the session by executing the command *Join Session*, which invokes the *connectToPartner* function of the *MessageManger*. This function fetches the host's WebSocket IP and then connects to it. Once a connection has been established, the *MessageManager* will open up a new WebSocket for the backend communication. Next, the *MessageManager* calls the *startSDKWorker* function to open a backend WebSocket, initializing the *SDKWorker* object. The backend process is modelled in detail in Figure 3.14, but is left out of Figure 3.11 to improve the readability. Once messages are received from a partner, the data is serialized

[12] `https://firebase.google.com/`

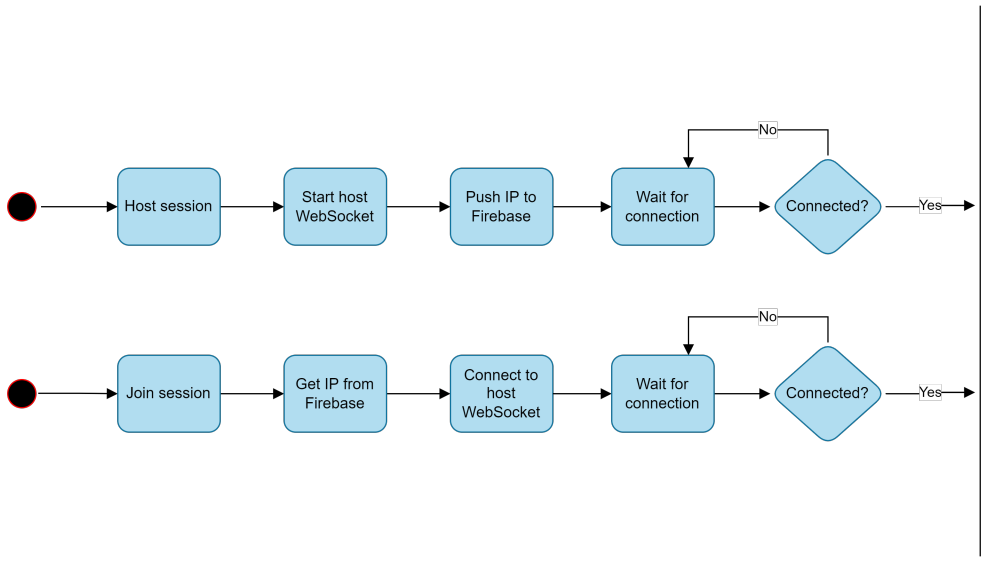


Figure 3.8: Process view (activity diagram) of joining and hosting a session.

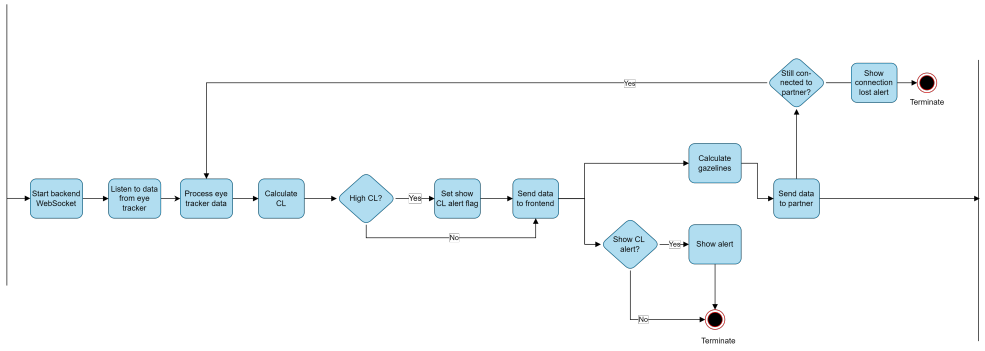


Figure 3.9: Process view (activity diagram) of reading data from eye tracker to sending data to partner. This process is continuation of Figure 3.8.

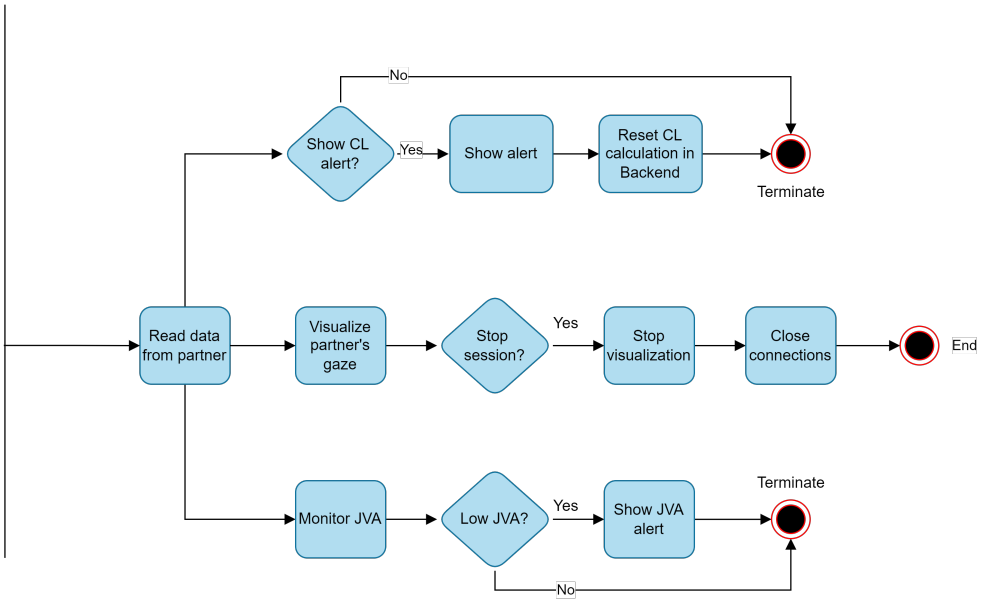


Figure 3.10: Process view (activity diagram) of reading data received from partner to stopping the session. This process is continuation of Figure 3.9.

and handled by the *GazeDataManager*.

Once a user wants to leave a session, the user will execute the *Stop Session* command. First, the command will close all connections which the *MessageManager* has open. Then, the *MessageManager* resets the *GazeDataManger* object, which involves removing any rendered visualization in VS Code and the moving average calculation.

Jumping To Partner

Figure 3.13 shows the process of *jumping to a partner*. First, the user issues the *Jump to Partner* command. Then, the *GazeDataManager* handles the action using the VS Code API function *revealRange*, which will scroll the user's document to the partner's gaze location.

Handling gaze data

Figure 3.14 illustrates how the frontend uses the data received from the backend to visualize gaze. The process is message-driven and starts when the backend sends data over a WebSocket to the frontend. The *MessageManger* will serialize the data and then convert the gaze coordinates to the corresponding line of the document opened in the editor. Then the data is sent to the partner through the

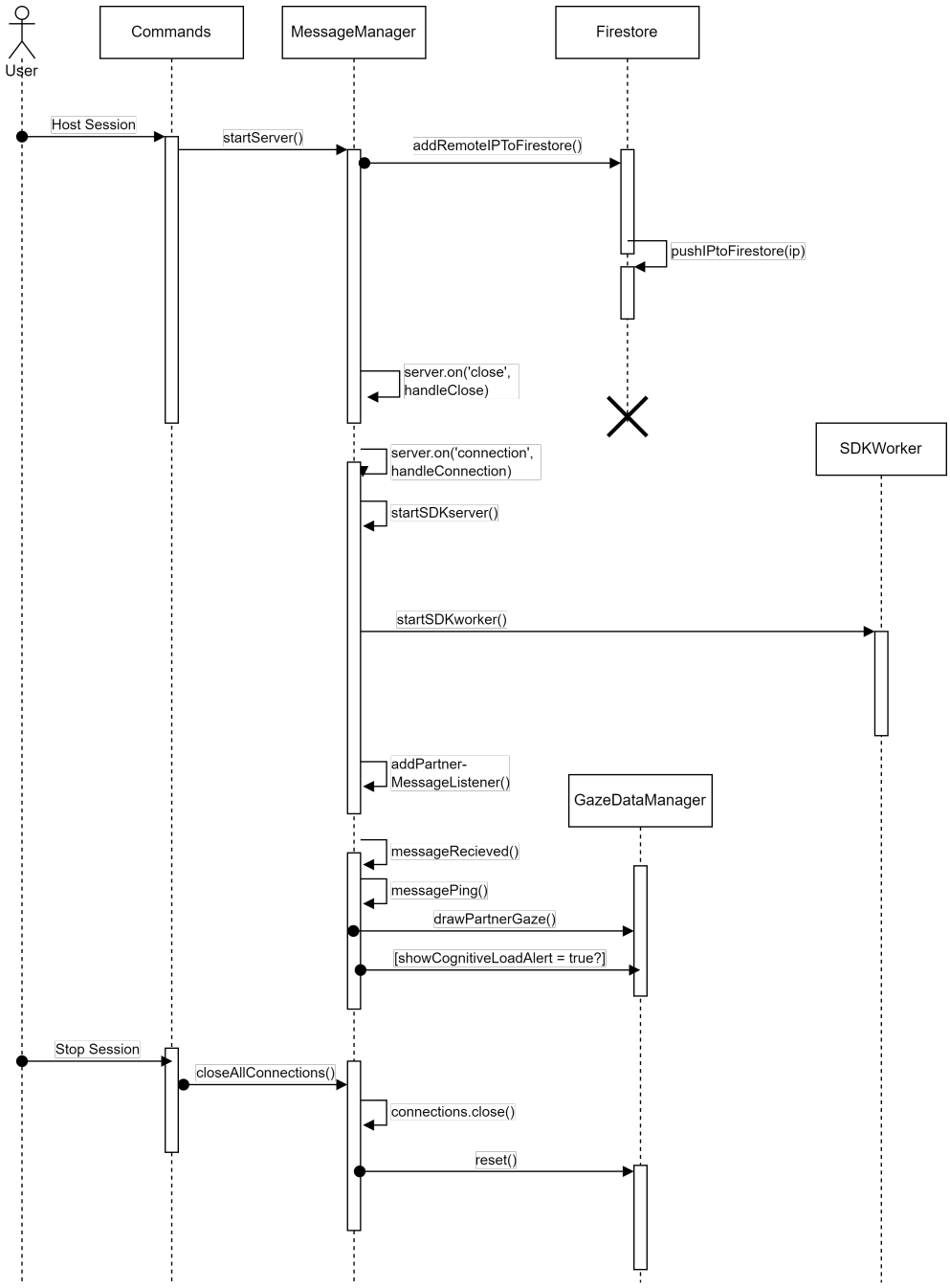


Figure 3.11: Process view (activity diagram) of reading data from eye tracker to sending data to partner.

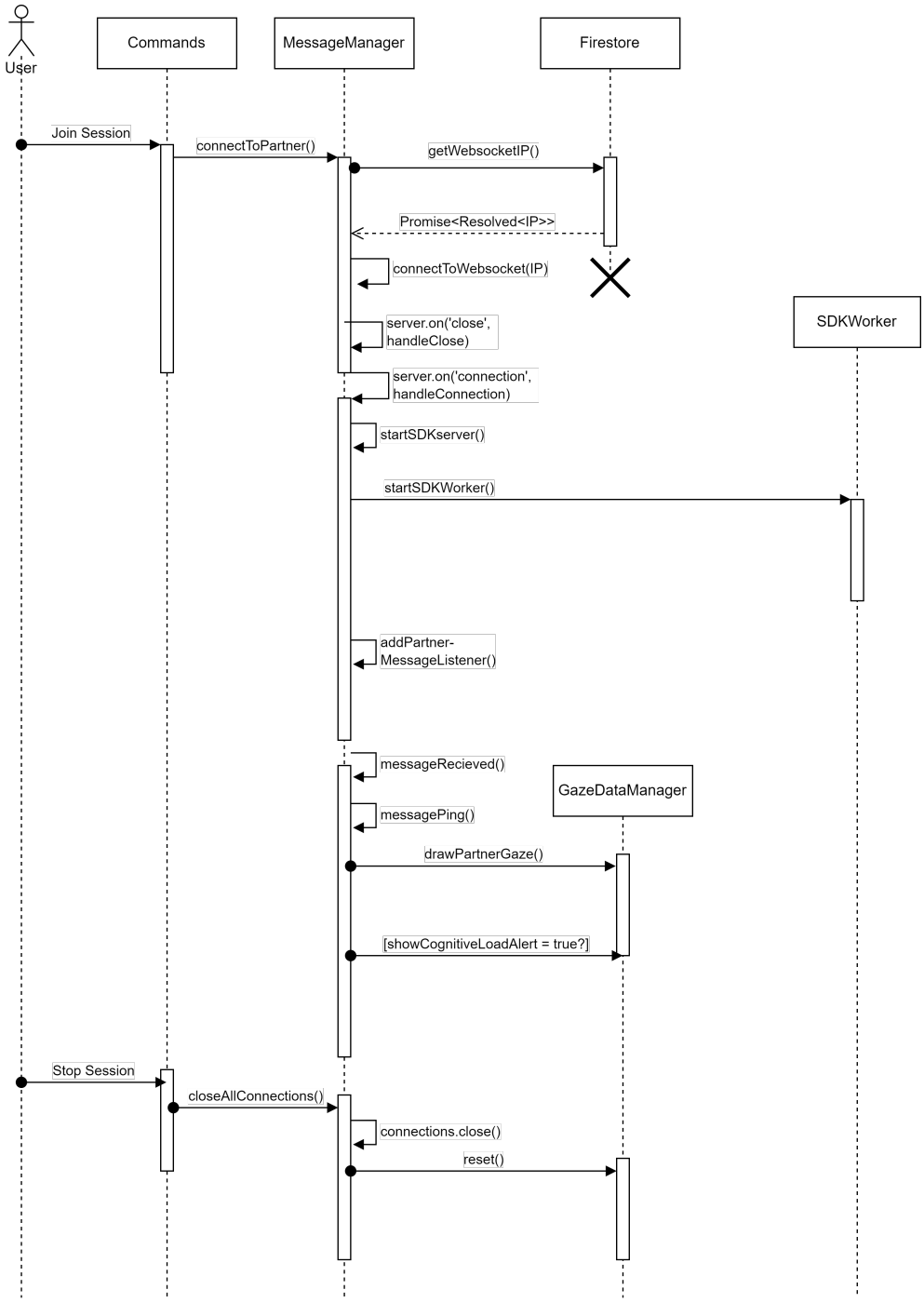


Figure 3.12: Process view (sequence diagram) of reading data from eye tracker to sending data to partner.

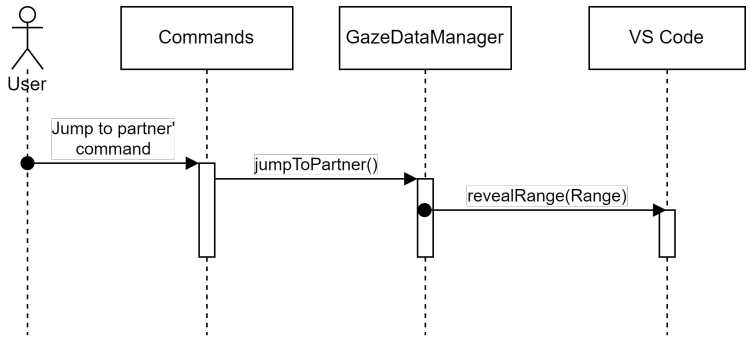


Figure 3.13: Process view (activity diagram) of reading data from eye tracker to sending data to partner.

partner connection. *MessageManager* checks if the backend has detected that a cognitive-load-based instructional alert must be shown due to high cognitive load. If the *showClAlert* flag is equal to true, the *showClAlert* function of the *Alert* object will be called. The object, will in turn, call the *showWarningMessage* function of the low-level VS Code API, thus giving the user instructional feedback.

When a message is received from the partner, the *MessageManager* will parse the data and use it to visualize the partner's gaze. It also checks if the partner has the *showClAlert* flag set. If the flag is set, an alert will be shown. Finally, when drawing the partner's gaze, the *GazeDataManager* will check if their gazes overlap, thus achieving JVA and giving behavioral feedback.

Calculating cognitive load

We implement Duchowski *et al.* [32]'s proposed cognitive-load metric Index of Pupillary Activity (IPA) for measuring a user's cognitive load. Figure 3.15 depicts the logic for calculating the cognitive load and how the system determines to give instructional feedback to the users with high cognitive load. This whole process is message-driven, meaning that for each gaze recording received from the eye tracker, which occurs with a 90 Hz frequency, cognitive load is calculated.

Calculating cognitive load in real-time has not been done in this manner. Previous studies have calculated cognitive load based on eye-tracking metrics, but only post-experiment [10]. Our feedback system incorporates real-time calculation of cognitive load and provides feedback based on this measurement. Thus, we experimented with using a rolling window of pupil diameter values. The rolling window stores pupil diameters from the last two minutes and the values are used to calculate the moment-to-moment IPA value. Using a rolling window mitigates abnormal IPA values caused by wrong readings from the eye tracker and avoids

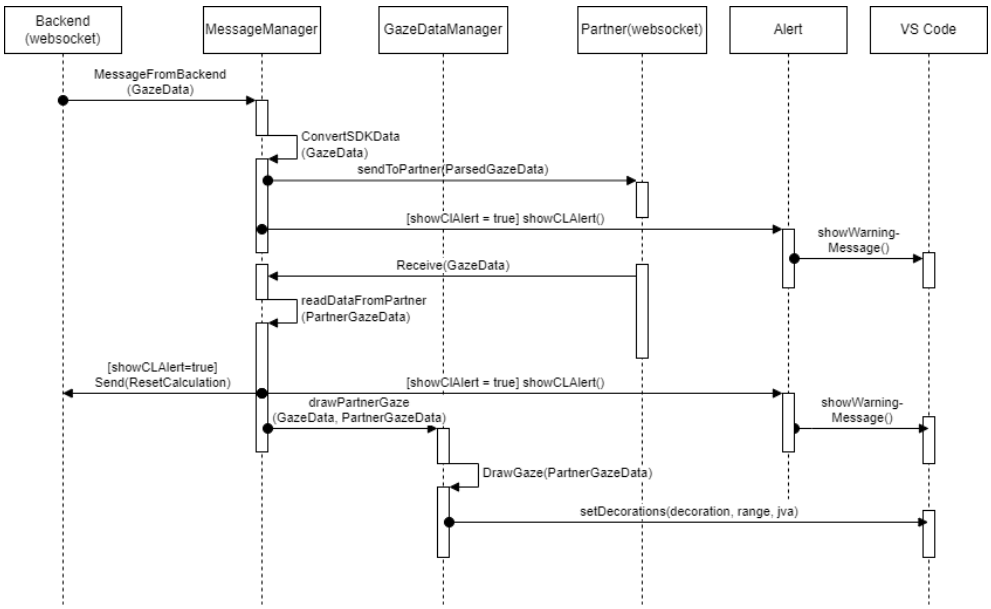


Figure 3.14: Process view (activity diagram) of reading data from eye tracker to sending data to partner.

outliers.

The system calculates an average IPA value using the moment-to-moment values from the last two minutes to trigger cognitive-load alerts. The value is compared to the moment-to-moment value of IPA. The user is cognitively overloaded when the percentage change of moment-to-moment IPA is above 18% compared to the average IPA for at least 20 seconds. The 18% threshold was selected based on the results from our pilot tests, which showed that a minimum 18% increase for at least 20 seconds fitted for the pilot testers' cognitive load. The system sets an alert flag to true in the backend, triggering a cognitive-load alert for both participants. The average IPA value is compared against a threshold for determining when the user has cognitive overload.

Following the algorithm laid out by Duchowski *et al.* [32], the system removes gaze data 200 ms before and after a blink. A blink occurs when both eyes have invalid coordinates, i.e., negative or missing coordinate values. In addition, the average IPA value is reset to the value it had 200 ms earlier. The system stores the timestamp for when a blink occurred and checks if 200 ms has passed before continuing to store pupil values in the rolling window list.

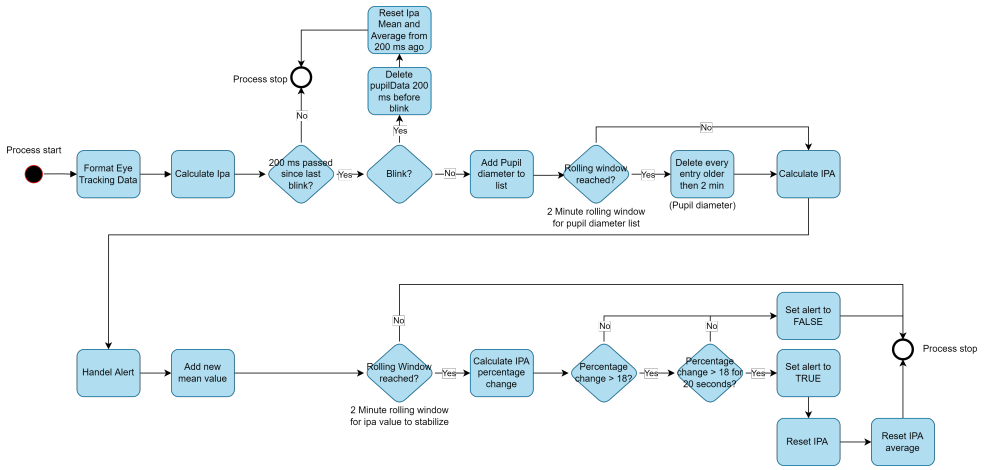


Figure 3.15: Process view of how the system calculates cognitive load based on pupil diameter and how it handle alerts.

3.2.3 Development View

The development view is shown in Figure 3.16. The view is represented as a package diagram and shows how ETS's source code is organized from a developer's point of view. The diagram is divided into two modules, the backend and the frontend module, corresponding to the source code's organization. The code is structured in compliance with the *package by feature* convention. The convention places classes that constitute a feature together in the same package. This is done to increase cohesion and decrease coupling.

The backend module contains packages which facilitate communication between the eye tracker and the feedback system. In addition, it handles everything related to the calculation of cognitive load and when to trigger alerts.

The frontend module facilitates communication between the two partners and manages the data received from the backend. The *root* package contains the main file, the commands component which users can run and various configurations such as shortcut definitions. The *connection* package maintains all connections, i.e., connection to a partner, connection to the backend and connection to the third-party service Firestore. *GazeData*'s primary purpose is to transform the gaze coordinates into the corresponding line in the source code document opened in VS Code's editor window. It also handles the logic for determining when the partners achieve JVA. Finally, the *visualization* package contains every component related to the user interface. These components determine the gaze visualization style, how alerts are shown and which buttons to render in the user interface.

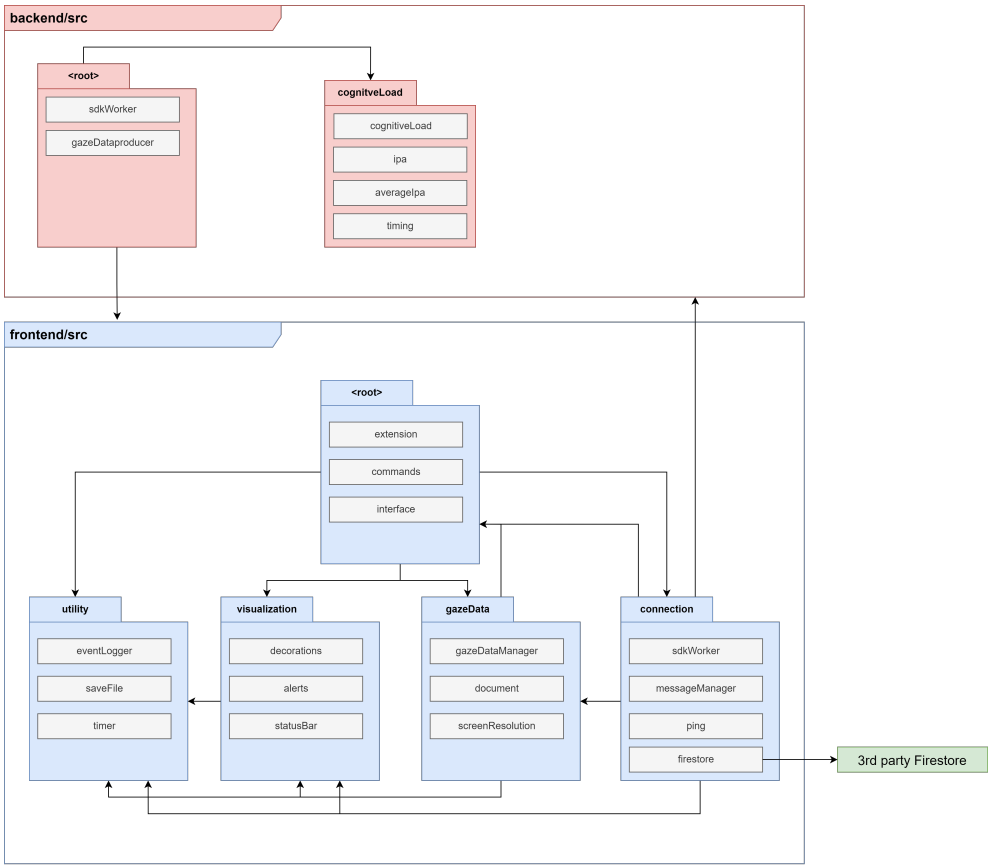


Figure 3.16: Development view of the system's components.

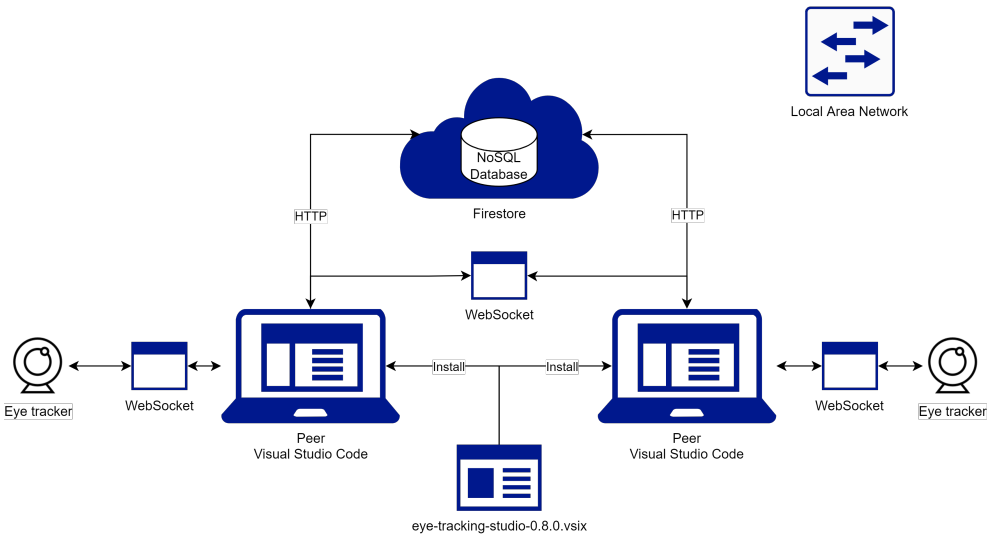


Figure 3.17: Physical view which depicts the topology of software components on the physical layer.

3.2.4 Physical View

Figure 3.17 depicts the system's topology of software and physical components and the connections between them. The ETS extension is manually installed on both peers' VS Code instances from a *vsix*-file, a file format used to distribute VS Code extensions. Two peers running the extension are required to set up a RPP session. When a peer starts a session in ETS, it communicates with Firestore's[13] SQLite database over HTTP to serve and store IP addresses for WebSockets. When a session is established between the two peers, they both communicate with their respective Tobii 4C eye trackers over a WebSocket object. The peer who hosts the session creates a WebSocket server which facilitates communication between the two. The ETS extension requires that both peers are connected to the same Local Area Network (LAN) or to a Virtual Private Network (VPN) connection which allows peer-to-peer communication.

3.3 Data analysis

Since our study is a mixed-method research study, we divide the data analysis into two main sections; *Quantitative* and *Qualitative*. This section describes how the data collected will be presented in the results section, what kind of data we analysed, how it was processed and analysed and which statistical tests we used

[13] `https://firebase.google.com/docs/firestore`

for the statistical inference.

3.3.1 Quantitative

Our quantitative data analysis focuses on *performance*, *cognitive load* and *joint visual attention (JVA)*. The analysis aims to answer our research questions. We rely on two metrics logged from every experiment to analyse performance: the number of bugs fixed and the time spent solving the tasks. After each experiment, we noted how many bugs were fixed and gave them 1 point for each bug solved. In addition, time spent solving the tasks for each condition was recorded. During initial pilot testing, we observed that the Tetris task, which was used under the feedback condition (FC), was more complex than the task given during the control condition. Based on the feedback from the pilot test participants, we felt confident that the Tetris task's scores could be multiplied by 1.5 to account for the difference in difficulty.

We use the Pearson's correlations test to check for correlation between the *pre-test score* and performance (Score & time on task). To evaluate the median score values between the two conditions, we use Wilcoxon non-parametric test.

The non-parametric two-samples Wilcoxon test is used to evaluate the median cognitive load values between the two conditions. To evaluate the change in cognitive load, we compute mean IPA for three-time frames surrounding an alert. The first period uses the data collected two minutes before an alert. The second period is defined as the time from an alert was shown to two minutes after the alert. Finally, the third period starts where the second period ends (two minutes after an alert was received) and ends two minutes later. These groups are called *CL before* (the two minutes before an alert), *CL during* (the two minutes after an alert) and *CL after* (the period ranging from two minutes to four minutes after an alert). Welch One-way ANOVA is performed to evaluate if the IPA values differ for the three periods.

We follow D'Angelo and Begel [10]'s definition of JVA, i.e., a pair achieves JVA when their gazes overlap for at least 100 ms. We compute the amount of time each pair spent in JVA based on timestamps and which lines of code they looked at from our log data collected during the experiments.

Because the length of the source code for each experiment condition is different, we multiply the JVA score for the FC by 1.75. The chance of two people simultaneously looking at the same line is $\frac{1}{2^n}$, where n is the number of source lines of code. The Snake game has 128 source lines of code, whereas the Tetris game has 224. Therefore, there is a probability of $\frac{1}{2^{128}}$ of looking at the same line at any time during the task in the NFC and $\frac{1}{2^{224}}$ for the FC. Because the code in

the FC is vertically longer than the code in the NFC, it is less likely to achieve JVA during the feedback task. To mitigate this task length discrepancy, we multiply the amount of JVA achieved during FC based on the code length ratio between the two games ($\frac{128}{224} = 1.75$). Wilcoxon non-parametric test is used to evaluate the median for each condition.

All statistical tests use a p-value of less than .05 to indicate a statistically significant difference.

3.3.2 Qualitative

We conduct a qualitative data analysis of the semi-structured interviews of the participants:

1. We began transcribing the interviews' audio recordings once we had conducted all interviews.
2. We performed a thematic content analysis of the data using the inductive coding approach. To aid with the data coding, we used NVivo[14], an application for performing qualitative data analysis. Once we had coded every transcript with the open coding approach, we then utilized axial and selective coding to analyze for overarching themes and connections in the interviews.
3. We read through the transcripts one more time to check if we had missed any crucial codes.
4. We counted the frequency of the number of pairs which touched upon each theme before refining a final set of themes and concepts.

To ensure the rigour of our analysis, we triangulate findings against the results of the quantitative analysis [62]. Additionally, we highlight and present any contradictory evidence found. Lastly, we use constant comparison when identifying common themes in the interviews [63]. To convey the credibility of our findings, we present our results using illustrations of concept trees, where we present the themes as concepts and the number of pairs who commented on the theme. Also, we present quotes from the participants and any quotes that contradict our overarching themes. We only present the most poignant results.

[14] `https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home`

Chapter 4

Results

4.1 Quantitative results

We present the analysis results of the within-subjects design experiment in the following sections. Every pair was exposed to two conditions: the control condition called the no-feedback condition (NFC), and the feedback condition (FC).

4.1.1 Performance

The analysis showed a significant difference in task completion time between the NFC and the FC. Task completion time was significantly less in the control condition ($t(37) = -3.11, p = .004$). We allocated 20 minutes for each condition, and in the FC, every pair used the entire allocated time. Six pairs completed the entire debugging task before the given time limit in the control condition. We computed the Pearson correlation coefficient to assess the linear relationship between the pretest score and the task completion time for the control condition. There was a negative correlation between the two variables ($r(38) = -2.6, p = .012$). There was no evidence of a linear relationship between the CAQ scores and bugs fixed, and time on task. CAQ provided an *enjoyment* and *motivation* score for each participant and we computed Pearson correlation test between each metric:

- Enjoyment Score & bugs fixed in feedback task: ($r(38) = 1.1, p = .27$)
- Motivation Score & bugs fixed in feedback task: ($r(38) = 0.16, p = .87$)
- Enjoyment Score & bugs fixed in control task: ($r(38) = 0.74, p = .47$)
- Motivation Score & bugs fixed in control task: ($r(38) = -1.31, p = .19$)

Pearson's correlation test showed no evidence of a correlation between the pretest score and the bugs fixed score in either NFC or FC. To test for normality in the data, we ran the Shapiro-Wilk test which reports a significant p-value for both

scores: *Snake score:* ($W = 0.75, p < 0.05$) and *Tetris score:* ($W = 0.46, p < 0.05$). The scores for each condition showed evidence of non-normality, and therefore, we conducted the non-parametric Wilcoxon rank-sum test to compare the two independent score variables. The median score for the control condition was 4.5 ($IQR = 3$), whereas the median score in the feedback condition was 3 ($IQR = 0$). The Wilcoxon test showed that there was a significant difference between the score between the two conditions ($W = 954, p = 0.02, n = 80,$ effect size $r =$ $.362(moderate)$).

4.1.2 Cognitive Load

To evaluate the effect the feedback system had on cognitive load, we analysed the median values of the IPA scores for both conditions. The data was not normally distributed, and therefore we used the non-parametric two-samples Wilcoxon test. The median IPA score for the control condition was .096 ($IQR = .053$), whereas the median in the feedback condition was .084 ($IQR = .037$). The Wilcoxon test showed no evidence of a significant difference in IPA scores between the conditions. ($W = 807, p = 0.38,$ effect size $r = 0.101$) as shown in figure Figure 4.1.

As described in Figure 3.15, the feedback system calculates cognitive load in real-time and gives feedback when the cognitive load is too high. During the experiment, participants who got cognitive-load-based feedback had the option of receiving a hint to reduce their cognitive load. During the 20 experiments, the system gave 31 alerts in the FC. In 20 of the 31 alerts, the participants were willing to receive a hint to solve the task at hand. To evaluate the impact timely hints had on their cognitive load, we define three two-minute periods from which we estimate their cognitive load (IPA scores), described in Section 3.3.

The QQ plot shown in Figure 4.2 visualizes how all the points fall approximately along the reference line. We can therefore assume normality in the data. This is further supported by the Shapiro-Wilk test, which reports a non-significant p-value ($p = 0.08$). Levene's test indicated unequal variances ($F = 5.09, p = .0008$). Since the homogeneity of variance cannot be assumed, we compute Welch One-way Anova since it does not require the assumption of equal variances.

Welch One-way Anova was performed to evaluate if the cognitive load (CL) differed for the three groups: *CL before* ($n = 40$), *CL during* ($n = 40$), *CL after* ($n = 40$). Results showed that cognitive load values was statistically significantly different between groups ($F(2, 69) = 3.34, p = 0.041$). The Games-Howell post hoc test revealed that the increase from *CL during* to *CL after* (-0.04, 95% CI (-0.07 to -0.001)) was statistically significant ($p = 0.04$), but none of the other groups had a significant difference between them. Results from the post hoc test is visualized

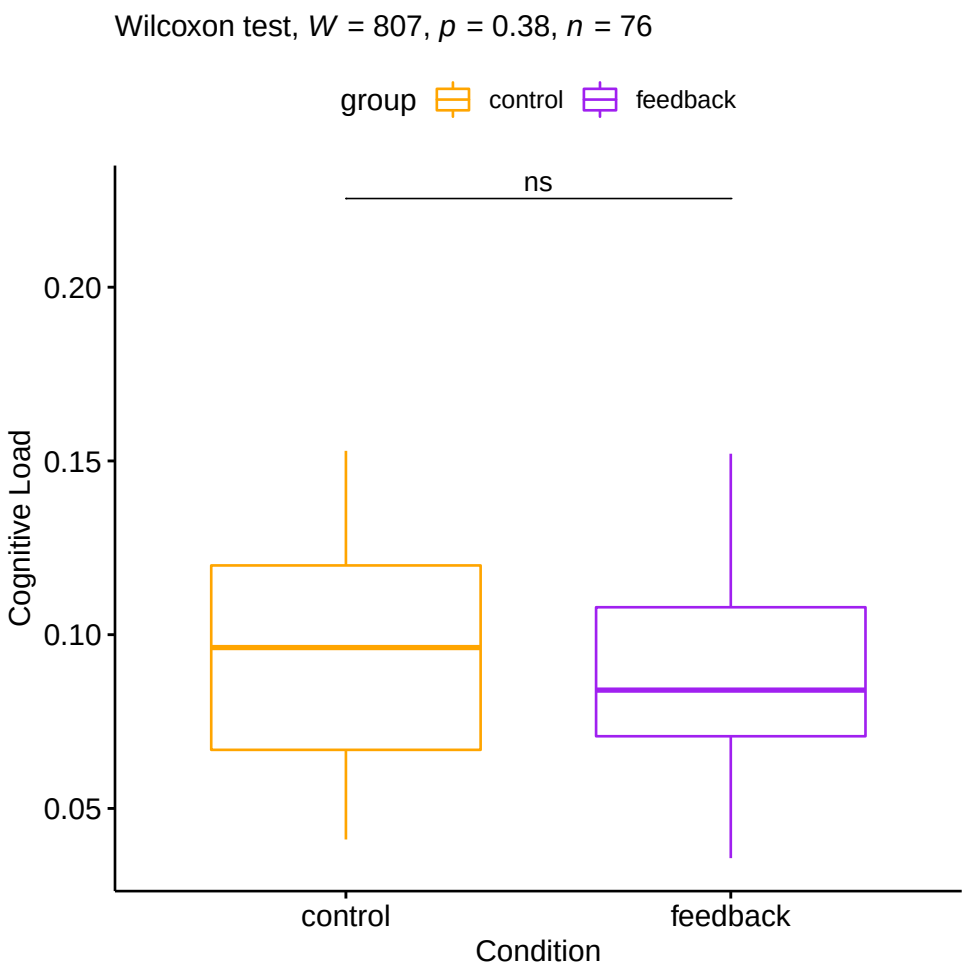


Figure 4.1: Box plot showing mean ipa values for each condition.

in Figure 4.3.

4.1.3 Joint Visual attention

By following the method described in D'Angelo and Begel [10], we counted the total duration of a pair looking at the same source code area for estimating their JVA. The gaze overlaps that lasted less than 100 ms were filtered out because they are likely caused by spurious eye movement. As described in Section 3.1.2, the feedback system visualizes the gaze with a nine-line high visualization. We add four lines on each side of the participant's gaze coordinates on the vertical axis. Therefore, we define the occurrence of JVA as *when the pair's gazes overlapped with one line inside the gaze range of nine lines by more than 100 ms.* The amount

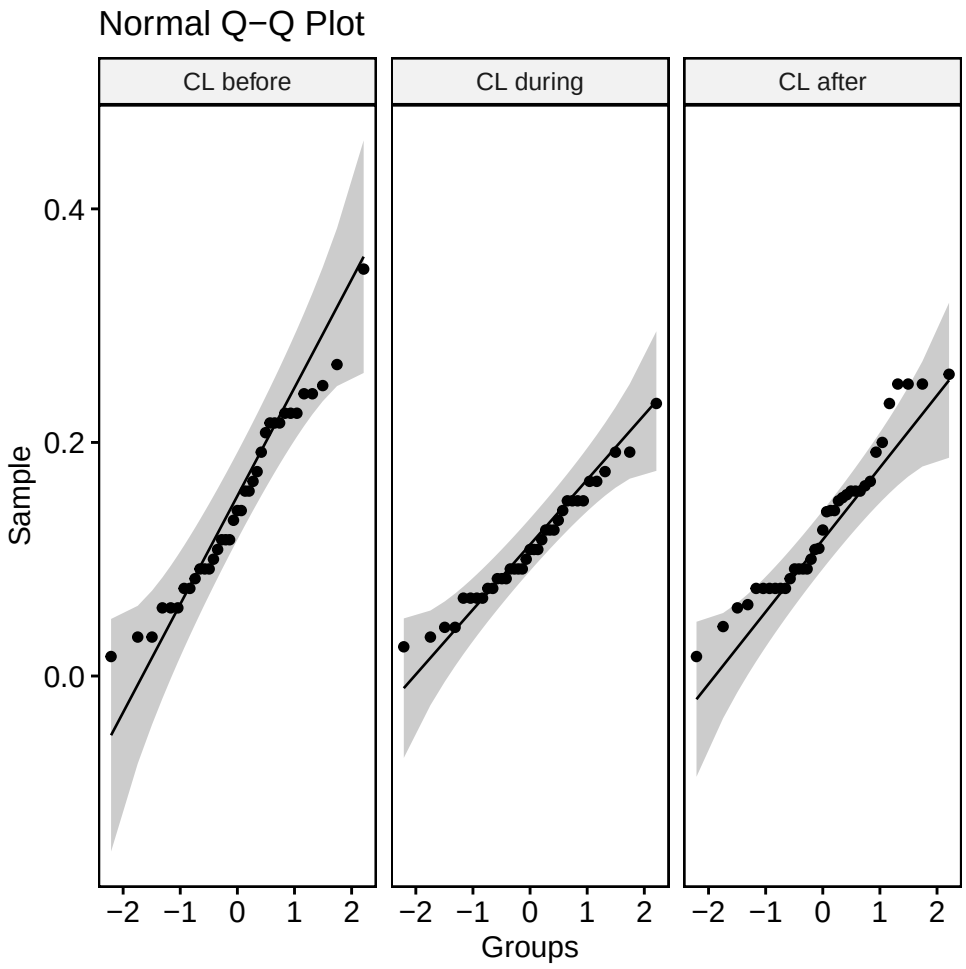


Figure 4.2: QQ plot showing the correlation between the cognitive load in each group and the normal distribution.

of JVA of every pair for each condition is shown in figure Figure 4.4.

Because of our small sample size, we performed a Shapiro Wilk test to see if our data was normally distributed and to choose an appropriate statistical method. The distributions were significantly non-normal for the *control* task ($W = 210, p = 0.04$), but the *feedback* task did not show evidence of non-normality ($W = 0,97, p = 0.83$). As we want to compare the two scores, which come from the same participants subject to both conditions, we performed the Wilcoxon signed-rank test because it does not assume normality in the data.

The Wilcoxon signed-rank test was used to compare the gaze visualization's effect on the pair's amount of time spent in JVA. The median *percentage of JVA* in the control group was 28% ($IQE = 0.049$) and 47% for the feedback group. The

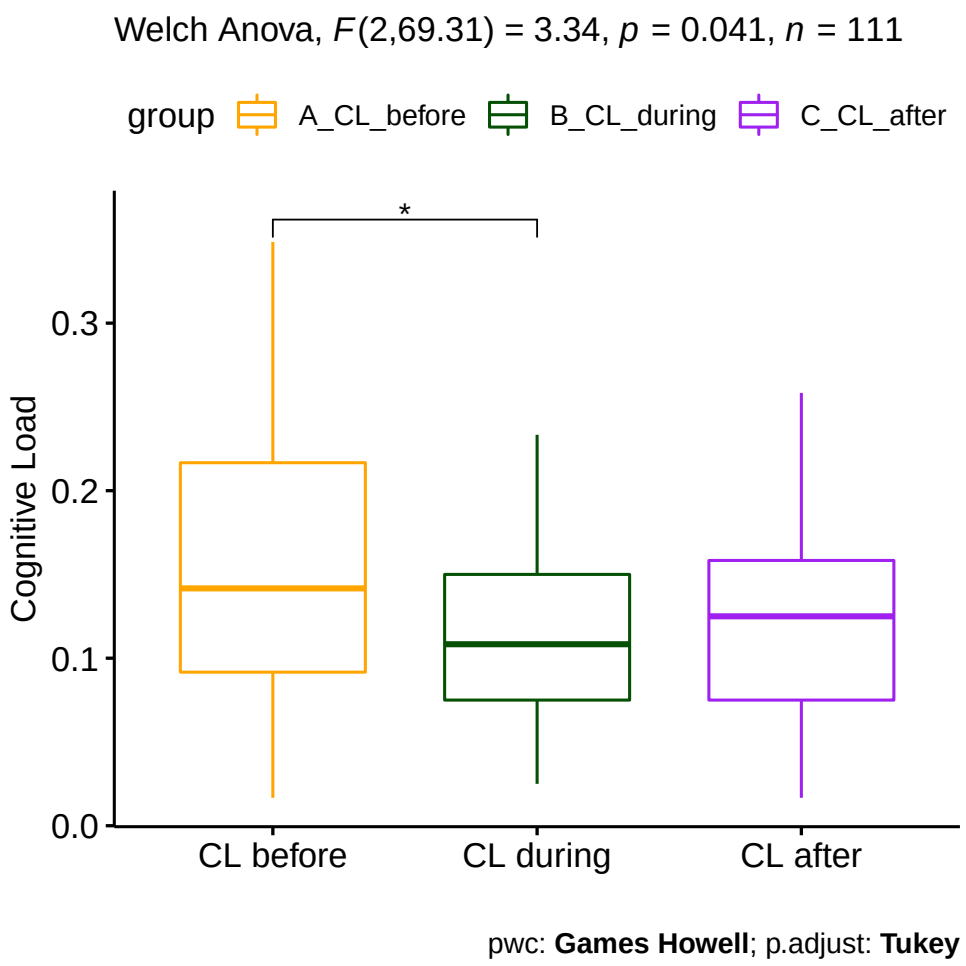


Figure 4.3: Box plot showing mean cognitive load values in the three groups.

test showed that gaze visualization elicits a statistically significant change in the amount of JVA ($W = 54, p = 0.0001$).

JVA alerts, as shown in Figure 3.2, were given if the pair's gazes did not overlap within a minute. Only six JVA alerts were triggered in total. Results show that the mean percentage of JVA before an alert was 9%, and after an alert, it was 12,7%. Because of the low sample size, we did not perform any statistical tests.

4.2 Qualitative results

From the interviews, we identified several themes. The most prevalent themes are summarised in Table 4.1. The following subsections present our findings categorised by features and themes. Quotations are taken from interview transcripts.

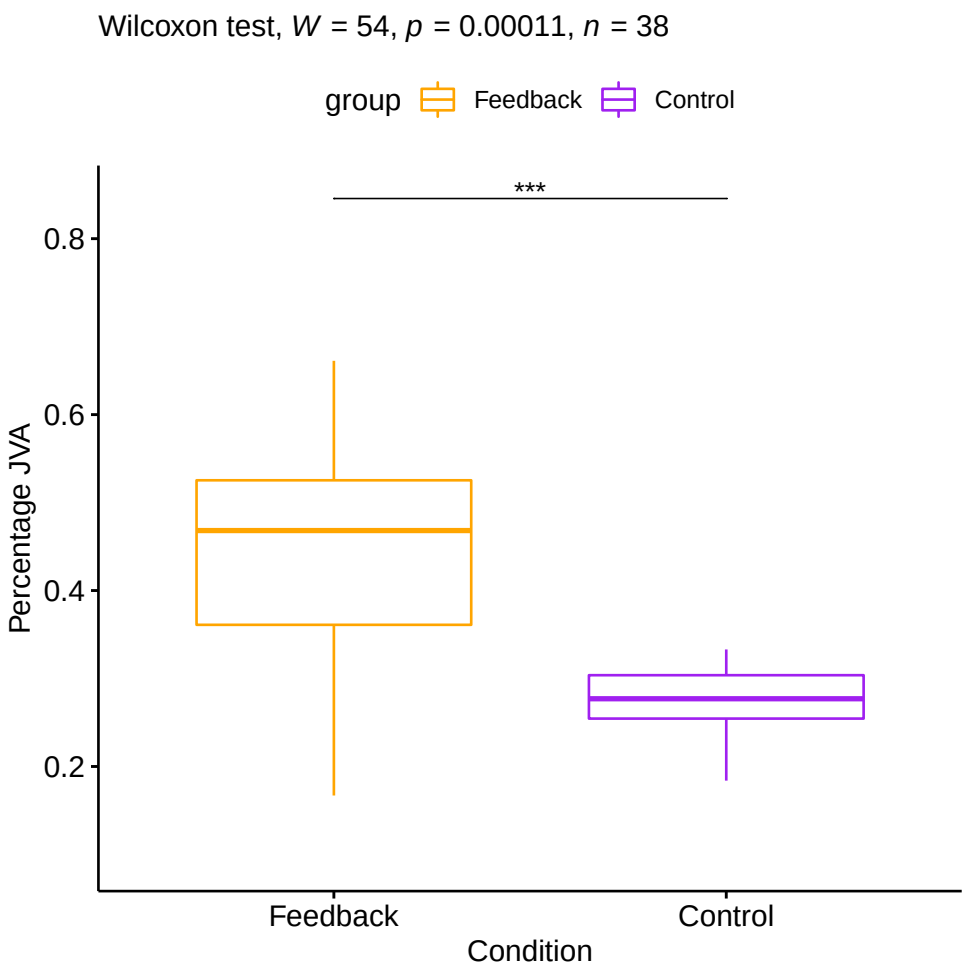


Figure 4.4: Box plot of the amount of gaze overlaps between the two experimental conditions.

4.2.1 Alerts

The cognitive load and JVA alerts, FR07 and FR08, respectively, received mixed reviews from the participants. The mixed reviews are based on the 18 pairs who discussed the alert feature in the interview. The results are summarised in Figure 4.5. It shows that ten pairs had positive comments and 12 pairs had negative comments, which means that each pair found both advantages and disadvantages with the feature. The participants who did not like the alert feature said that the alerts were intrusive or not timely. The participants felt that the alerts were intrusive since they would lose focus and their train of thought when concentrating on a piece of code.

Table 4.1: Main themes identified from the interviews.

Theme	Topic
Useful	Feedback system
	Gaze visualization (FR04 and FR05)
	JVA visualization (FR06)
	Jump to Partner (FR09)
	Alerts (FR07 and FR08)
Improves communication	Gaze visualization
	Jump to Partner
Intrusive	Alerts

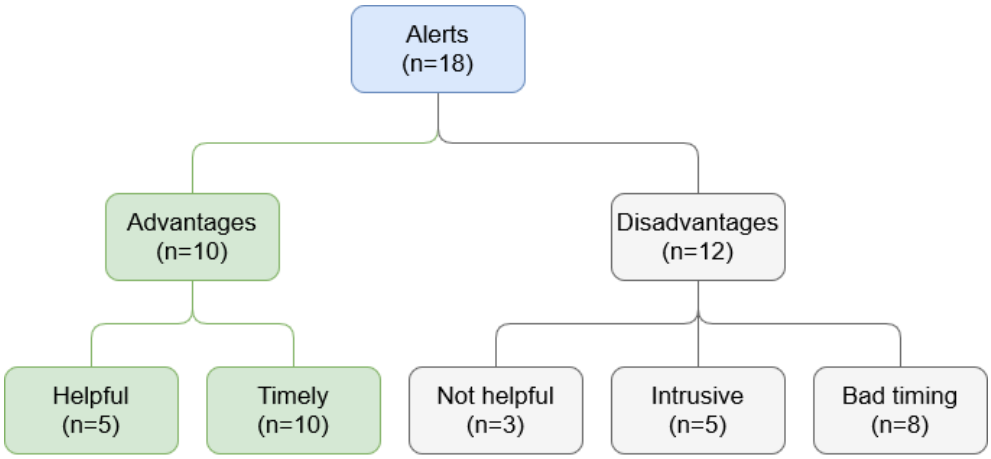


Figure 4.5: Concept tree showing answers from the interview when participants discussed alerts. *n* is the number of pairs commenting on the concept.

> *I'm not quite sure about those alerts, they were very "inn your face". They really pulled me out of the flow I was in. In some situations it may be a desired effect, but I experienced it mostly as disturbing, at least in the current form factor.*
>
> – Pair 16, participant B.

The pairs did also not agree on the alerts being timely. Ten pairs said that they were on time and accurate regarding them being stuck in the code or not knowing how to proceed. One pair even said that 'The hint alert was spot on. Came at the right moment.' – *Pair 20, participant 20.* On the other hand, eight pairs state that the timing was off. There was no clear tendency if the pairs thought

the alerts appeared too early or too late during the task. It was evident that the pairs thought that the Tetris task was much more complicated than the Snake task. Some pairs stated they felt they could not utilize our feedback system to its full extent since they had to spend much effort understanding the source code rather than taking advantage of the system. Interestingly, two pairs said that the FC task was so difficult that any time would have been a good time to receive a cognitive-load-based alert and a hint.

> *I think the timing of those warnings could have fit at any time really. Struggled a lot with the Tetris code so it was appropriate to get notice at any time. Or I do not know completely. I did not think much about what time it came based on how we collaborated or how we looked at things.*
>
> – Pair 10, participant B.

> *It fit all the time [alert timing], we had no idea what we were doing. (...) I think we struggled all the time in a way. So in that sense, it fit no matter when it came up.*
>
> – Pair 13, participant A.

Even though the alerts were intrusive, pairs generally thought that the alerts were helpful. For example, pair 16, who said that the alerts were intrusive, said that the alert timing was fitting with them being stuck or struggling with a bug.

> *I actually feel that the timing was quite good. But the last time there, I thought that if I had had 10 more seconds I might have figured out the bug myself, but I can not say for sure it just felt that way.*
>
> – Pair 16, participant B.

> *For my part, if I had sat alone, then it would have fit pretty okay. I pondered a lot myself and understood a bit of the code but I thought I needed help to get one step further. At least I could move on faster if I had taken the tip when I was offered it.*
>
> – Pair 16, participant A.

> *It was useful, a bit like that it managed to understand somehow now we're not making any progress here. A link to that load somehow.*
>
> – Pair 18, participant A.

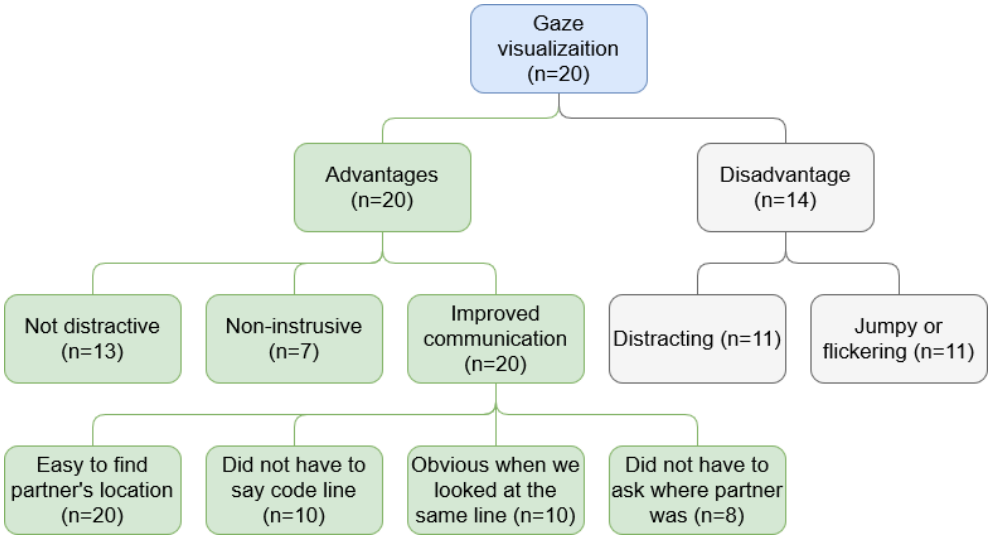


Figure 4.6: Concept tree showing answers from the interview when participants discussed the gaze visualizations. *n* is number of pairs commenting on the concept. Only the themes with the highest frequency are showed.

4.2.2 Gaze visualization

Generally, the pairs said that the gaze visualization was helpful. Even though there were mixed reviews on the visualization being distracting, as shown in Figure 4.6, all pairs agreed that the visualization was more helpful than distracting. Initially, two participants thought the visualizations were more distracting than helpful. However, one of them changed their mind during the interview and said they needed to get used to the workflow. Not one pair toggled the gaze off (FR10), but one pair forgot about the feature. Incidentally, that pair complained about the visualizations being distracting and even said they wished they could turn it off. Five pairs argued that the system's visual noise got reduced over time and that they would have gotten used to it if they had the chance to get more familiar with the system.

> *I think maybe it was almost a bit distracting sometimes when you see that side scroll bar go up and down all the time and it caught my eye quite a few times maybe. For better or worse.*
>
> – Pair 9, participant B.

> *Very subtle way of seeing where the others were without it dominating, very cleverly made really, very cool. Seemed like it was pretty accurate and pretty responsive.*

– Pair 2, participant B.

[Talking about the gaze visualization] I think it was so little that it was not disturbing. I think it was okay to have it in the side view so that I can look at in the scroll bar and see how the code is read [by the partner] and then I know how I can work based on it. – Pair 10, participant B.

The pairs would sometimes disagree if they thought the visualization was distracting or not, which was evident in the comments from pair 20, who said:

Like, the gaze tracking, I found it a bit distracting and because (...) I tried to focus on what I was thinking, but then I saw her just [indicates going up and down with waving hand] to look at some other stuff and then I just thought OK so I should look at that too, yeah? So but, like maybe, it could be better if I just got to work with it some more perhaps. It's not a way of working that I'm used to, so. Yeah, but at the moment I found it distracting rather than helpful.

– Pair 20, participant B.

I found it really helpful. Yeah, I didn't have this distraction thing at all, 'cause I usually have the problem to figure out where the other person is looking and so with that it was really easy to find and figure it out. And I don't I didn't mind seeing that he was looking at something else, 'cause it's normal and we both tried to figure stuff out. When she started talking, I could just go to where she is and knew what she was talking about, so I found it really helpful.

– Pair 20, participant A.

As Figure 4.6 shows, every pair said it was easy to find their partner's location in the code, and eight pairs mentioned that they did not have to ask where their partner was located in the source code document. However, two pairs said they needed to ask where their partner was and say precisely which code line they were investigating. Those two pairs said that the visualization was not accurate enough to precisely indicate the line they were working on. Eight pairs mentioned that the visualization could have been more precise by making it smaller. However, they realized that there is a trade-off with having a smaller visualization since it would be more distracting. However, several pairs had improvement suggestions for making the visualizations more precise. The most prominent of those suggestions was that they could precisely indicate which line they were referring

to by clicking on that line. Thus, the cursor would show the position to the partner. However, one cannot rely on the cursor visualization by itself, as the cursor will not always be at the location of a programmer's attention [64]. This is also described by pair 14:

> *I think it would have been a good combo if you see where the other person clicks, because you know you are in the same place so then you can just press and mark here [with the cursor] (...), because then you know that they have not clicked in an location and then scrolled all the way down in the document because I have experienced it many times. You're not where the mouse pointer is. So when you say "Yes, here" you answer "oh, but where are you?".*
>
> – Pair 14, participant A.

4.2.3 Communication and cooperation

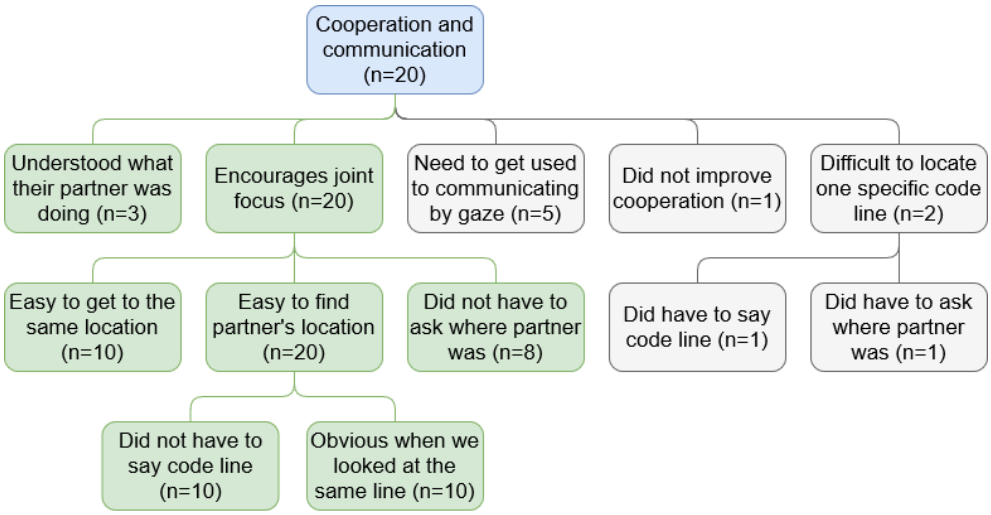


Figure 4.7: Concept tree showing pairs' comments in regards to cooperation and communication.

We identified a consensus among the pairs that the communication during pair programming was easier and more efficient with the feedback system. Seeing each partner's gaze location made it easier to communicate code locations, making communication more efficient for the pairs. As shown in Figure 4.7, the pairs found that they did not have to ask where in the code their partner was or which code line they were working on. Referential grounding was also made more

efficient by the *Jump to partner*-feature (FR09), which was the most liked feature along with the shared gaze feature.

> *I used the eye tracking a bit. I think it was very nice really because I could sit and look at a completely different place when she says "Look here" before she says the line. Then I press [jump to partner] and I see where she is looking.*
>
> – Pair 16, participant A.

> *I've done some remote pair programming before and it's a little annoying to have to say "Hey you, on line 136 there is one thing". Didn't have to do it so many times here, was pretty nice.*
>
> – Pair 16, participant B.

> *Yes, then you do not have to say, like we had to do a lot at the end here, like "that line one hundred and six", so you can just do alt+p so yes, it is something.*
>
> – Pair 19, participant B.

However, two pairs said they still needed to be more precise when pointing to a location in the code. The gaze visualization indicated the area of their gazes but not the specific code line, especially when the code was hard to understand.

> *So it is still useful to be able to say "yes the one there on line 47" even if you see that you are looking at the same thing because it is not so precise, but then you can also just use the "click in the code" [refers to cursor visualization]*
>
> – Pair 14, participant A.

> *It works quite well, but as she said: when you start debugging difficult things then it still becomes "where are you now, really?".*
>
> – Pair 10, participant A.

However, several pairs mentioned that they could increase the system's benefits by getting more used to using the system and communicating by gaze. In retrospect, they recognized that they sometimes would say which lines they were working on even though they could communicate by gaze. The reasoning behind this was that they were accustomed to saying line numbers, and it was difficult to lay the habit behind.

> *I think I did it anyways [said which line of code they were working on] out of habit.*
>
> – Pair 12, participant A.

> *This system would have been incredibly fun to get used to, then there would be more mental communication instead.*
>
> – Pair 2, participant A.

> *It's good, but it's a limitation that we often go back to something like... It's a bit unusual, you can not completely get used to it in a way. So I fail to fully integrate it so easily in a 20 minute session. [What do you mean by going back to?] To say which code line I'm working on.*
>
> – Pair 14, participant B.

Nineteen out of 20 pairs stated that they preferred to debug with the feedback system. However, four pairs explicitly stated that they missed the feedback system during the control task—especially those who had the feedback task first noticed increased difficulty communicating location in the code. The following excerpts are from pairs who performed the FC before switching to the NFC.

> *(..) I noticed it a lot when we did not have it [the feedback system], and I had to ask like, "where are you?" then having to say line numbers and then it's like "and then I'll have to scroll there instead of just doing [imitates pressing short cut for jump to partner]". Then I don't have to ask, and I can move a lot faster.*
>
> – Pair 11, participant A.

> *At first I thought that when we turned it off [the feedback system] it didn't matter; I was not going to notice any difference, but I did. Because, when I saw where she was, it was much easier to "now we are in the same place, then we can talk about it" because otherwise I sit and look at a thing while I have no idea what she is looking at. That was what I noticed on the first task [FC] that «oh well now we are in the same place, then I can just talk about it" because then she understands what I'm talking about.*
>
> – Pair 13, participant A.

Another benefit of seeing the partner's gaze is that the pairs could infer how their partner looked at the code and how they processed it. If a partner's gaze

moved up and down the document, their partner would know that the other person was looking at the code as a whole to get an overview. This was especially prevalent at the beginning of the task, where the pairs had to familiarise themselves with the program. If the gaze was hovering around a limited area, the partner knew that the other partner was focusing a lot on that area which could mean that he had found something interesting. This meant that the pairs knew when they should look at the same area or not, e.g., if a partner found a bug and suddenly moved their attention to a new location, the other partner knew that he should follow along.

Pair 17 highlights the goal of gaze awareness: encouraging the pairs to join their focus of attention. When they were told about the JVA alert (FR07), they explained that they did not receive the alert because of the shared gaze:

> *Yeah, but that's also because just it being there [the gaze visualization] made me look more often at what he's looking at.* – Pair 17, Participant A

4.2.4 Adopting the feedback system

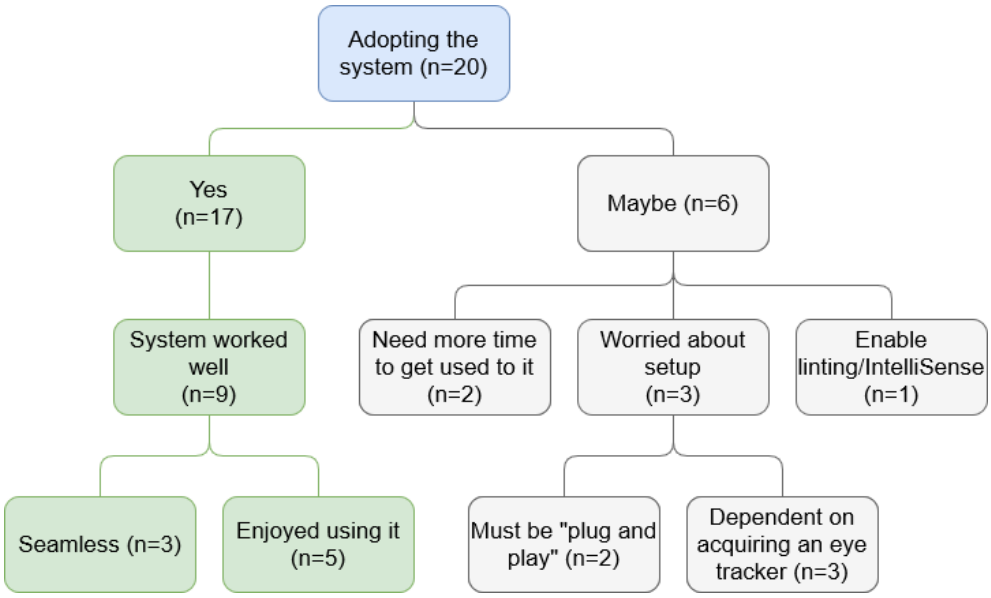


Figure 4.8: Concept tree showing answers from the interview when the pairs where asked if they would like to use the system in their day-to-day remote pair programming.

Regarding adopting the feedback system, the participants generally said they would like to use it in their day-to-day remote pair programming situations. The

findings are summarised in Figure 4.8, which shows that 17 pairs said they would use the system without any objections. Some of the recurring comments about the system were that it was "seamless and minimalistic" and that they enjoyed using it.

> Participant-B: As long as it is as stable as it was here, it would have been awesome.
>
> Participant-A: I completely agree with that. Also, in relation to how inconspicuous it was, it felt like you can always have it on, without it mattering, so you learn that «oh now you do not have to pay attention to...». Then you do not really need to turn it off, like now. It would have been nice to get rid of that "Which line?" or "Where are you now?".
>
> – *Pair 2.*

No one said they would not use the system. However, six pairs had some concerns about the system before being willing to adopt it. For instance, the interviews revealed that the system should be "plug and play" and be compatible with other programming tools, i.e. requiring IntelliSense[1] support (intelligent code completion) and linting. Other comments were about the need to acquire an eye tracker and its cost. However, once we told them the eye tracker's cost, they thought it was affordable. In addition, two pairs said that they would use the system if a web camera could replace the eye tracker.

The participants felt that our system could alleviate many of their issues regarding remote pair programming. Additionally, every pair said they preferred to debug with the feedback system *on* rather than *off*. Many pairs commented that they had spent much time recently in remote pair programming sessions using a screen sharing setup using online meeting video services. Several pairs suggested that they gladly adopt the system's gaze sharing aspect, but not the alert features. They suggested that the alerts could be more appropriate in other programming scenarios, such as in classrooms or online lecturing, where the teacher can monitor and identify struggling students.

> Yes. We wrote a bachelor together and then we used "share screen" on Discord. It was a bit like... that you had to find out where you are and stuff, go up and down and stuff, and say where you are going. This would have been very useful then. – *Pair 13, participant B.*

[1] `https://code.visualstudio.com/docs/editor/intellisense`

Chapter 5

Discussion

This study explores the design and use of a feedback system in a remote pair programming (RPP) context and how it impacts collaboration, performance and cognitive load. To our knowledge, Eye Tracking Studio (ETS) is a first of its kind feedback system. Previous work evaluates how to apply shared gaze in a RPP context, but our thesis is the first to provide cognitive-load-based instructional feedback. We made a feedback system that recorded pair programmers' cognitive load unobtrusively using affordable and ubiquitous eye trackers. Furthermore, the participants preferred to debug with our feedback system enabled in the feedback condition (FC) rather than not having it as in the no-feedback condition (NFC). The key findings related to our research questions are:

- Instructional feedback had no impact on the performance of remote pair programmers.
- Performance was significantly higher in the no-feedback condition (NFC) than for the feedback condition (FC).
- No evidence of a significant difference in cognitive load when using the feedback system.
- Results indicate a significant decrease in cognitive load right after the feedback system proposes a helpful hint (instructional feedback).
- Results indicate that visualising each partner's gaze (behavioral feedback) significantly increases the amount of achieved joint visual attention (JVA).

In the following subsections, we interpret and evaluate the impact of our results. Additionally, we discuss limitations and recommend what future research should take away from our results.

5.1 Interpretation and Implications of the results

5.1.1 Performance

Our research goal was to investigate how timely hints triggered by pair programmers' cognitive load and gaze awareness impact the performance of remote pair programmers. Our results indicate that the feedback system does not improve performance. We saw a lower median debugging score and higher time on task for the task in the FC than the NFC, rejecting hypothesis H1. Results indicate no evidence of a correlation between the pretest score and the pairs' debugging score. The pretest score indicates the participants' program comprehension and prior knowledge of the Python programming language. We did not see an increased amount of bugs fixed for those with higher pretest scores. This could be caused by the discrepancy in task difficulty or because the pretest focused more on Python syntax, while the task during the experiment was more tied to game logic. There was a negative correlation between the pretest score and time on task for the NFC, which could indicate that their prior knowledge affected their performance for the task in the NFC condition.

After artificially increasing the score for the task in the FC, we still see a higher score in the NFC task, which further supports that the task difficulties are skewed. The participants also verified the task difficulty skewness in the interviews. Out of the 20 pairs, only two solved more than two bugs out of the six bugs present in the FC source code. Three pairs solved less than two bugs, resulting in 15 out of 20 pairs solving two bugs during the FC.

One of the bugs was very game-breaking in the FC, and it was tough to find the solution. Most pairs opted to solve this bug first and spent much time solving it. However, 19 out of 20 pairs needed two hints before they could solve this particular bug. Once they had solved it, most of the allocated task time was spent, which hindered a lot of the participants in solving the other bugs. Pairs commented on being stuck on one bug for a long time, which is unfortunate because they did not get to utilize the feedback system to its full extent. Participants said in the interviews that they had to spend a lot of effort on understanding the source code rather than taking advantage of the system. However, the difficulty was accounted for by weighting each bug solved higher, and therefore, the results show that the feedback system does not affect performance in an RPP setting. The task difficulty should have been balanced between the two conditions. We suspect that the very complex task negatively affects the results because it is unlikely to see a performance increase. Additionally, artificially increasing the score by 1.5 should ideally have been avoided.

Feedback systems utilizing shared gaze have been shown to correlate with and improve the performance score of collaborative problem-solving tasks, and remote search tasks [45, 46]. Although, as discussed, our result did not show any indication of performance improvement, this could be a result of the discrepancy in task difficulty. However, D'Angelo and Begel [10] also reported no significant difference in task completion time in an RPP setting. Therefore, the context in which feedback is provided should be investigated in other settings to evaluate the impact feedback has on performance.

5.1.2 Cognitive load

The use of cognitive load and the Index of Pupillary Activity (IPA) metric for giving real-time feedback in an RPP context is novel. Therefore, we evaluate the feedback's effect on the programmer's cognitive load.

Our goal was to decrease programmers' cognitive load when they had cognitive overload by giving them instructional feedback in the form of hints for solving the bugs. The median cognitive load value was higher for the NFC than the FC, but not statistically significant and therefore rejecting hypothesis H2. As previously discussed, the FC had a considerably more complex task than the NFC, but this is not reflected in the programmers' cognitive load. Furthermore, the IPA value used for measuring cognitive load is tightly coupled with task difficulty [32]. However, we did not see the expected result of an increased IPA value for the FC task even though the task was complex. This could indicate that our feedback system reduced the cognitive load imposed by the instructional design because we expect that the task difficulty would impose more cognitive load than the snake task.

Our feedback system proposes a method of triggering cognitive load based feedback in real-time based on an IPA value threshold. The participants gave mixed reviews of the timing of the alerts. Some pairs felt that the alert was timely, while others felt they were either triggered too late or fitting at any time. This indicates that the cognitive load threshold for triggering alerts could be more accurate and refined. Therefore, and because the triggering method is novel, it requires further research to evaluate the appropriate threshold for triggering instructional feedback based on cognitive load.

When the cognitive load based alerts were triggered, the participants had the option to receive hints. During the FC, the system gave 31 alerts to the participants. In 65% of the alerts, participants accepted to receive a hint. This does not mean that 35% of the alerts participants did not feel the need for a hint because some pairs said they did not want a hint because of their pride in solving the tasks without any help. We saw a correlation between the pre-test and the number

of hints each participant received. This could indicate that those with less prior knowledge found the task more difficult and needed more hints. These results also suggest that the feedback system managed to detect the users who needed more help than those who had more prior knowledge and help them to continue with the task.

Results described in Section 4.1.2 and Figure 4.3 indicate that receiving cognitive load based feedback and a hint significantly decreases cognitive load momentarily. This implies that the hints are helpful. However, because the cognitive load decrease was only temporary, the results may also indicate that the participants stepped *out* of the task to focus on the hint instead. This is further supported by the qualitative analysis results where participants felt that the alerts made them lose focus and their train of thought. This could also explain why cognitive load decreased between *CL before* and *CL during*. Future work should investigate how to better deliver instructional feedback without causing participants to lose focus on the task at hand.

5.1.3 Joint Visual Attention

One of the issues of working spatially distributed is becoming aware of a partner's focus of attention. We set out to explore a feedback system for improving the collaboration and coordination of remote pair programmers. Our results show that ETS contributes to solving this issue – JVA increased from the NFC to the FC and participants said the visualization made it easy to locate their partner's attention supporting hypothesis H1. This is supported by the findings of our qualitative analysis, where the participants found it much easier to align their visual focus and identify when they were looking in the same location. Prior work has found the same results, where the real-time visualization of partner's gaze has increased the amount of JVA pairs achieve [10, 28, 45, 49].

ETS makes it easier for pairs to talk about source code since they can be confident that they are discussing the same piece of code. Therefore, collaboration improved since they did not have to spend as much effort with referential communication when discussing a location in the source code – our feedback system encouraged joint focus. However, we did not see a significant decrease in cognitive load between the two conditions, which can mean two things. First, even though gaze awareness was thought to reduce the cognitive load imposed by instructional information and the adverse effect of split attention, it did not. Second, the Tetris task was much more complex than the Snake task, so an increase in task difficulty overshadows a reduction in extraneous load caused by instructional design.

Our Jump to Partner feature (FR09) helped reduce time spent coordinating

joint focus. As several pairs commented, they used the jump to partner shortcut whenever their partner wanted them to *look here*. Additionally, this feature was one of the most liked features of the system. We, therefore, regard our feature implementation as successful. We recommend that future feedback systems implement a feature for automatically moving the viewport to the partner's location. We have not seen such a feature in previous research, and its use and effect should be evaluated further, e.g., using referential communication analysis.

On the other hand, we did not see an increase in performance between the two conditions even though the percentage JVA increased. This finding is not in line with the findings of Schneider and Pea [49], who saw that JVA was positively correlated with learning gain.

Based on behavioral feedback and the signaling principle, we managed to increase JVA in a minimal and non-intrusive manner. Participants did not think the gaze visualization caused too much distraction nor too much blinking when indicating JVA. We regard this feature as successful and advise that future feedback systems should also share gazes in real-time with an indication when gazes are aligned. Some pairs said that the visualization encouraged them to look where their partner was looking, meaning the signaling principle was employed successfully.

In addition to indicating JVA, the pairs received instructional feedback when their gaze had not overlapped during the last minute (FR07). This feedback was given as an alert and instructed the pairs to work more closely together so they would increase their amount of JVA. However, we did not find any evidence that JVA increased after receiving the alert. Only four pairs received the alert, which is too small a sample size for conducting statistical analysis. Because the number of pairs who received a JVA alert was so low, we can not determine the usefulness of this feature. Furthermore, when pairs are aware of each other's gazes, they are automatically encouraged to align their visual focus, making the JVA alert redundant. We argue that providing behavioral feedback in the form of gaze visualization is incentive enough for pair programmers to achieve JVA. Therefore, future feedback systems do not need an alert feature for increasing a pair's JVA.

5.1.4 Design of Feedback System

It is interesting to note that even though the participants had the option to toggle the gaze visualization off, no one did that. Even though some pairs complained that the gaze visualization could be distracting and jumpy as described in Figure 4.6. Therefore, the system only partially fulfils NRF02, because the benefits of the gaze visualization outweighed the drawbacks. However, the feature did have

a hot key like jump to partner had. Prior work has indicated a need for a feature to toggle gaze on and off [2, 18]. Therefore, if the functionality was more accessible, we probably would have seen more gaze toggling.

The effectiveness of the alerts where not so good. The alerts where designed to force an answer from the user before they could proceed with their task. The reasoning behind implementing the alerts in such a manner was that the pilot test showed that the alerts could be missed, the participants did not notice the alert, if they were displayed in the corner of the VS Code window. Additionally, we followed the notification guidelines from VS Code, which also states that if we require an answer form the user, we should use this type of alerts. However, many participants thought that this alert was too intrusive and they would lose their focus and their flow when the alert was shown. This was especially prevalent for pairs which received several alerts during the experiment. Therefore, the system does not fulfill NFR02. The alerts should not have been so intrusive.

5.2 Limitations and Future Work

Even though these contributions impact the design of feedback systems and remote pair programming, they come with limitations which must be addressed in future work.

We used multipliers in the data analysis to reduce the discrepancies in task complexity and the number of source lines of code between the no-feedback condition (NFC) and feedback condition (FC). Using multipliers is justified, although the generalizability of the findings is limited due to the artificial changes. Future research should ensure that the source codes are of similar length and complexity before conducting user testing.

There were mixed results on the alert based on cognitive load metrics. As described in Section 4.2.1, some participants felt that alerts were not timely and that they were too intrusive, which led to a loss of focus. The threshold and conditions for providing alerts based on cognitive load, as described in Section 3.2.2, has limited testing and need further research to establish more precise evaluations of when to trigger feedback. Further research regarding this topic is encouraged with in-depth pilot testing to do determine more data for determining thresholds.

Results from the quantitative analysis show that many of the participants experienced the cognitive load alerts as intrusive. When alerts appeared, we provided hints on paper, which led the participants to look away from the screen. Looking away from the screen means that the eye tracker cannot record gaze data, which affects the cognitive load calculation. Future work using cognitive-load-

based feedback is encouraged to provide feedback that does not require participants to look away from the screen. Furthermore, our participants suggested that the hint-based instructional feedback could be more appropriate in a teacher-learner context. In addition, several participants would rather use our feedback system's behavioural feedback than the instructional feedback in real-world RPP. Therefore, we recommend future research to evaluate real-time cognitive-load-based feedback in other contexts, such as students learning to program.

As described in Section 4.2.2, five pairs argued that they would have gotten more used to the visualization of gaze if they had more time to get familiar with it. Each programming session lasted a maximum of 20 minutes, and this could be argued not to be adequate time for someone to get used to new visual input. More time should have been allocated, which is supported by participants who said that the visual noise was reduced over time. We recommend that future research lengthen the experiment or let participants become more familiar with the system beforehand.

One of our contributions is developing a feedback system based on the relatively affordable Tobii Eye Tracker 4C. This model was released in 2016 and has since been replaced by the improved Tobii Eye Tracker 5 [44]. Unfortunately, the older eye tracker has a lower gaze data recording frequency and field of view than the improved Tobii Eye Tracker 5, and we acknowledge that this limits the eye-tracking quality [42, 43]. Therefore, with better algorithms for head tracking and more support for head movement, future research is encouraged to use the newer model because it will provide higher data quality.

5.3 Generalizability of Results

There are some reasons why our results are not generalizable. Firstly, we applied self-selection sampling when recruiting participants for the experiment. However, we had to turn to convenience sampling because the sign-up rate was so low that we needed to change strategy to get the required amount of participants. Unfortunately, we should have avoided convenience sampling should because it limits generalizations to the wider population [60]. We recruited 20 pairs, where the majority of participants were either students or alumni from NTNU. This population is too small to conclude that our findings generalize to a larger population.

Additionally, participants could sign up together with someone they knew to make it easier to recruit participants. This resulted in a sample where the majority knew each other beforehand, and some had pair programmed together before. The pairs who knew each other had an advantage because they were more com-

fortable with each other. Unfamiliar pairs might feel that it is more challenging to cooperate than the familiar pairs. This might affect our results in that the familiarity levels of the peers within a pair could have helped them manage the communication better than two strangers. Familiarity could also impact the generalizability for newly formed RPP pairs.

The generalizability of our finding where JVA is increased is limited. Due to the Tetris code being longer (having more lines of code) than the Snake code, we had to inflate the JVA score accordingly. However, our qualitative results show that the system helps users increase their joint focus.

5.3.1 Rigour in the Qualitative Data Analysis

Interpreting findings from the interviews using qualitative data analysis is arguably more subjective than quantitative analysis of qualitative data. Qualitative data can be interpreted differently by different researchers, and the issue of the validity of qualitative analysis has no definitive solution [65]. Therefore, there is a possibility that we have introduced bias to the results by not retaining an objective view of the data. However, we did implement countermeasures to avoid bias, such as triangulating the results against the quantitative analysis's results, highlighting contradictory evidence, presenting quotes from the interviews and following the well-established framework of constant comparison for analysing the data. Nevertheless, we could still have implemented additional efforts to ensure validity, such as respondent validation and peer review [65]. Therefore, there is a possibility that the qualitative results might be biased, but we did implement several countermeasures to avoid this. As a result, we believe we have kept bias at a minimum.

Chapter 6

Conclusion

In this thesis, we present and evaluate a novel feedback system developed as a Visual Studio Code (VS Code) extension for alleviating the limitations of remote pair programming (RPP). Referential communication in RPP is more challenging than in co-located pair programming because of the lack of non-verbal cues. Therefore, we investigate the feedback's impact on remote pair programmers' collaboration, performance and cognitive load. We design and implement a feedback system that provides twofold real-time feedback: gaze awareness and cognitive load management using inexpensive eye trackers and VS Code's extendability capabilities. Gaze awareness enables pair programmers to see their partner's attention in the editor, while the cognitive-load-based feedback aims to lower the task imposed cognitive load.

We evaluate the feedback system with a within-design experiment, where pairs worked together in a simulated RPP environment – with and without the help of our extension.

We contribute to technical knowledge by designing and developing a feedback system as a Visual Studio Code (VS Code) extension for remote pair programming (RPP). This feedback system uses inexpensive eye trackers to share gaze and estimate cognitive load in real-time. Providing instructional feedback based on real-time cognitive load is novel. The gaze visualization is found to be non-intrusive. Lastly, a new feature called *Jump to Partner*, which moves the editor view to the partner's gaze location in the document, is presented.

Our scientific contributions include proof that gaze awareness significantly increases joint visual attention (JVA) and improves referential communication. Additionally, instructional feedback decreased cognitive load momentarily and we recommend future work to find other less intrusive ways of giving cognitive load based feedback. Lastly, the feedback system showed no evidence of improving the

performance of remote pair programmers.

The lack of physical affordances in remote collaborative programming can hinder programmers from reaping the benefits of pair programming. However, using inexpensive eye-tracking technology, some of these physical affordances can be provided in a remote work environment using Eye Tracking Studio (ETS).

Bibliography

[1] L. Yang, D. Holtz, S. Jaffe, S. Suri, S. Sinha, J. Weston, C. Joyce, N. Shah, K. Sherman, B. Hecht and J. Teevan, 'The effects of remote work on collaboration among information workers,' *Nat Hum Behav*, vol. 6, no. 1, pp. 43–54, Jan. 2022, Number: 1 Publisher: Nature Publishing Group, ISSN: 2397-3374. DOI: 10.1038/s41562-021-01196-4. [Online]. Available: https://www.nature.com/articles/s41562-021-01196-4 (visited on 02/06/2022).

[2] Y. Zhang, K. Pfeuffer, M. K. Chong, J. Alexander, A. Bulling and H. Gellersen, 'Look together: Using gaze for assisting co-located collaborative search,' *Personal and Ubiquitous Computing*, vol. 21, no. 1, pp. 173–186, 2017, ISSN: 1617-4909.

[3] J. Sweller, 'Cognitive load theory, learning difficulty, and instructional design,' *Learning and Instruction*, vol. 4, no. 4, pp. 295–312, 1st Jan. 1994, ISSN: 0959-4752. DOI: 10.1016/0959-4752(94)90003-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0959475294900035 (visited on 19/05/2022).

[4] C.-Y. Tsai, Y.-F. Yang and C.-K. Chang, 'Cognitive load comparison of traditional and distributed pair programming on visual programming language,' in *2015 International Conference of Educational Innovation through Technology (EITT)*, Oct. 2015, pp. 143–146. DOI: 10.1109/EITT.2015.37.

[5] S. Pietinen, R. Bednarik and M. Tukiainen, 'Shared visual attention in collaborative programming: A descriptive analysis,' in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '10, New York, NY, USA: Association for Computing Machinery, 2nd May 2010, pp. 21–24, ISBN: 978-1-60558-966-4. DOI: 10.1145/1833310.1833314. [Online]. Available: https://doi.org/10.1145/1833310.1833314 (visited on 09/11/2021).

[6] Z. Sharafi, B. Sharif, Y.-G. Guéhéneuc, A. Begel, R. Bednarik and M. Crosby, 'A practical guide on conducting eye tracking studies in software engineering,' *Empirical Software Engineering*, vol. 25, no. 5, pp. 3128–3174, 1st Sep. 2020, ISSN: 1573-7616. DOI: 10.1007/s10664-020-09829-4. [Online]. Available: https://doi.org/10.1007/s10664-020-09829-4.

[7] G. A. Miller, 'The magical number seven, plus or minus two: Some limits on our capacity for processing information,' *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956, Place: US Publisher: American Psychological Association, ISSN: 1939-1471. DOI: 10.1037/h0043158.

[8] P. Chandler and J. Sweller, 'The split-attention effect as a factor in the design of instruction,' *British Journal of Educational Psychology*, vol. 62, no. 2, pp. 233–246, 1992, ISSN: 2044-8279. DOI: 10.1111/j.2044-8279.1992.tb01017.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8279.1992.tb01017.x (visited on 03/06/2022).

[9] S. Kalyuga, P. Chandler and J. Sweller, 'Managing split-attention and redundancy in multimedia instruction,' *Applied Cognitive Psychology*, vol. 13, no. 4, pp. 351–371, 1999, ISSN: 1099-0720. DOI: 10.1002/(SICI)1099-0720(199908)13:4<351::AID-ACP589>3.0.CO;2-6. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-0720%28199908%2913%3A4%3C351%3A%3AAID-ACP589%3E3.0.CO%3B2-6 (visited on 03/06/2022).

[10] S. D'Angelo and A. Begel, 'Improving communication between pair programmers using shared gaze awareness,' in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, New York, NY, USA: Association for Computing Machinery, 2nd May 2017, pp. 6245–6290, ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025573. [Online]. Available: https://doi.org/10.1145/3025453.3025573 (visited on 10/11/2021).

[11] S. D'Angelo and D. Gergle, 'Gazed and confused: Understanding and designing shared gaze for remote collaboration,' presented at the Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, California, USA: Association for Computing Machinery, 2016, pp. 2492–2496. DOI: 10.1145/2858036.2858499. [Online]. Available: https://doi.org/10.1145/2858036.2858499.

[12] R. Müller, J. R. Helmert, S. Pannasch and B. M. Velichkovsky, 'Gaze transfer in remote cooperation: Is it always helpful to see what your partner is

attending to?' *Quarterly Journal of Experimental Psychology*, vol. 66, no. 7, pp. 1302–1316, 2013. DOI: 10.1080/17470218.2012.737813. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1080/17470218.2012.737813.

[13] K. Higuch, R. Yonetani and Y. Sato, 'Can eye help you? effects of visualizing eye fixations on remote collaboration scenarios for physical tasks,' presented at the Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, 2016, pp. 5180–5190.

[14] S. E. Brennan, X. Chen, C. A. Dickinson, M. B. Neider and G. J. Zelinsky, 'Coordinating cognition: The costs and benefitsof shared gaze during collaborative search,' *Cognition*, vol. 106, no. 3, pp. 1465–1477, 1st Mar. 2008, ISSN: 0010-0277. DOI: 10.1016/j.cognition.2007.05.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010027707001448.

[15] K. Gupta, G. A. Lee and M. Billinghurst, 'Do you see what i see? the effect of gaze tracking on task space remote collaboration,' *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2413–2422, 2016, ISSN: 1941-0506. DOI: 10.1109/TVCG.2016.2593778.

[16] D. Akkil, B. Thankachan and P. Isokoski, 'I see what you see: Gaze awareness in mobile video collaboration,' presented at the Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, Warsaw, Poland: Association for Computing Machinery, 2018, Article 32. DOI: 10.1145/3204493.3204542. [Online]. Available: https://doi.org/10.1145/3204493.3204542.

[17] D. Akkil, J. M. James, P. Isokoski and J. Kangas, 'GazeTorch: Enabling gaze awareness in collaborative physical tasks,' presented at the Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, San Jose, California, USA: Association for Computing Machinery, 2016, pp. 1151–1158. DOI: 10.1145/2851581.2892459. [Online]. Available: https://doi.org/10.1145/2851581.2892459.

[18] B. Maurer, M. Lankes, B. Stiglbauer and M. Tscheligi, 'EyeCo: Effects of shared gaze on social presence in an online cooperative game,' S. Kriglstein, H. Hlavacs, R. Malaka, A. Lugmayr and H.-S. Yang, Eds., red. by G. Wallner, ser. Entertainment Computing - ICEC 2016, Cham: Springer International Publishing, 2016, pp. 102–114, ISBN: 978-3-319-46100-7.

[19] C. Liu, D. Kay and J. Y. Chai, 'Awareness of partners eye gaze in situated referential grounding: An empirical study,' presented at the 2nd Workshop on Eye Gaze in Intelligent Human Machine Interaction, 2011, p. 33.

[20] R. Stein and S. E. Brennan, 'Another person's eye gaze as a cue in solving programming problems,' presented at the Proceedings of the 6th international conference on Multimodal interfaces, State College, PA, USA: Association for Computing Machinery, 2004, pp. 9–15. DOI: 10.1145/1027933.1027936. [Online]. Available: https://doi.org/10.1145/1027933.1027936.

[21] C. Wohlin, 'Guidelines for snowballing in systematic literature studies and a replication in software engineering,' in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14, New York, NY, USA: Association for Computing Machinery, 13th May 2014, pp. 1–10, ISBN: 978-1-4503-2476-2. DOI: 10.1145/2601248.2601268. [Online]. Available: https://doi.org/10.1145/2601248.2601268 (visited on 13/12/2021).

[22] M. Tomasello, 'Joint attention as social cognition,' *Joint attention: Its origins and role in development*, vol. 103130, pp. 103–130, 1995.

[23] M. Cook, 'Gaze and mutual gaze in social encounters: How long—and when—we look others "in the eye" is one of the main signals in nonverbal communication,' *American Scientist*, vol. 65, no. 3, pp. 328–333, 1977, Publisher: Sigma Xi, The Scientific Research Society, ISSN: 0003-0996. [Online]. Available: https://www.jstor.org/stable/27847843 (visited on 23/04/2022).

[24] F. Broz, H. Lehmann, C. L. Nehaniv and K. Dautenhahn, 'Mutual gaze, personality, and familiarity: Dual eye-tracking during conversation,' in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, ISSN: 1944-9437, Sep. 2012, pp. 858–864. DOI: 10.1109/ROMAN.2012.6343859.

[25] M. Cherubini, M.-A. Nüssli and P. Dillenbourg, 'Deixis and gaze in collaborative work at a distance (over a shared map): A computational model to detect misunderstandings,' in *Proceedings of the 2008 symposium on Eye tracking research & applications*, ser. ETRA '08, New York, NY, USA: Association for Computing Machinery, 26th Mar. 2008, pp. 173–180, ISBN: 978-1-59593-982-1. DOI: 10.1145/1344471.1344515. [Online]. Available: https://doi.org/10.1145/1344471.1344515 (visited on 08/06/2022).

[26] D. C. Richardson and R. Dale, 'Looking to understand: The coupling between speakers' and listeners' eye movements and its relationship to discourse comprehension,' *Cognitive science*, vol. 29, no. 6, pp. 1045–1060, 2005, Publisher: Wiley Online Library, ISSN: 0364-0213.

[27] S. Whittaker and B. O'Conaill, 'The role of vision in face-to-face and mediated communication,' in *Video-mediated communication*, ser. Computers, cognition, and work, Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers, 1997, pp. 23–49, ISBN: 978-0-8058-2288-5.

[28] G. H. Kütt, K. Lee, E. Hardacre and A. Papoutsaki, 'Eye-write: Gaze sharing for collaborative writing,' in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: Association for Computing Machinery, 2nd May 2019, pp. 1–12, ISBN: 978-1-4503-5970-2. [Online]. Available: https://doi.org/10.1145/3290605.3300727 (visited on 10/11/2021).

[29] F. Paas, A. Renkl and J. Sweller, 'Cognitive load theory and instructional design: Recent developments,' *Educational Psychologist*, vol. 38, no. 1, pp. 1–4, 1st Jan. 2003, ISSN: 0046-1520. DOI: 10.1207/S15326985EP3801_1. [Online]. Available: https://doi.org/10.1207/S15326985EP3801_1 (visited on 19/05/2022).

[30] S. Kalyuga, 'Cognitive load theory: How many types of load does it really need?' *Educ Psychol Rev*, vol. 23, no. 1, pp. 1–19, 1st Mar. 2011, ISSN: 1573-336X. DOI: 10.1007/s10648-010-9150-7. [Online]. Available: https://doi.org/10.1007/s10648-010-9150-7 (visited on 05/05/2022).

[31] Y. Shi, E. Choi, R. Taib and F. Chen, 'Designing cognition-adaptive human–computer interface for mission-critical systems,' in *Information Systems Development: Towards a Service Provision Society*, G. A. Papadopoulos, W. Wojtkowski, G. Wojtkowski, S. Wrycza and J. Zupancic, Eds., Boston, MA: Springer US, 2010, pp. 111–119, ISBN: 978-0-387-84810-5. DOI: 10.1007/b137171_12. [Online]. Available: https://doi.org/10.1007/b137171_12 (visited on 07/06/2022).

[32] A. T. Duchowski, K. Krejtz, I. Krejtz, C. Biele, A. Niedzielska, P. Kiefer, M. Raubal and I. Giannopoulos, 'The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation,' in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, Montreal QC Canada: ACM, 21st Apr. 2018, pp. 1–13, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173856. [Online]. Available: https://dl.acm.org/doi/10.1145/3173574.3173856 (visited on 19/04/2022).

[33] J. Zagermann, U. Pfeil and H. Reiterer, 'Measuring cognitive load using eye tracking technology in visual computing,' in *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, ser. BELIV '16, New York, NY, USA: Association for Computing Machinery, 24th Oct. 2016, pp. 78–85, ISBN: 978-1-4503-4818-8. DOI: 10.1145/2993901.2993908. [Online]. Available: https://doi.org/10.1145/2993901.2993908 (visited on 08/06/2022).

[34] E. H. Hess and J. M. Polt, 'Pupil size in relation to mental activity during simple problem-solving,' *Science*, vol. 143, no. 3611, pp. 1190–1192, 1964, Publisher: American Association for the Advancement of Science, ISSN: 0036-8075.

[35] D. Kahneman and J. Beatty, 'Pupil diameter and load on memory,' *Science*, vol. 154, no. 3756, pp. 1583–1585, 1966, Publisher: American Association for the Advancement of Science, ISSN: 0036-8075.

[36] K. Krejtz, A. T. Duchowski, A. Niedzielska, C. Biele and I. Krejtz, 'Eye tracking cognitive load using pupil diameter and microsaccades with fixed gaze,' *PloS one*, vol. 13, no. 9, e0203629, 2018, Publisher: Public Library of Science San Francisco, CA USA, ISSN: 1932-6203.

[37] M. Schulte-Mecklenbeck, A. Kühberger and J. G. Johnson, 'A handbook of process tracing methods for decision research: A critical review and user's guide,' 2011, Publisher: Psychology Press, ISSN: 1135389780.

[38] A. F. Kramer, 'Physiological metrics of mental workload: A review of recent progress,' *Multiple-task performance*, pp. 279–328, 2020, Publisher: CRC Press, ISSN: 1003069444.

[39] J.-L. Kruger, E. Hefer and G. Matthew, 'Measuring the impact of subtitles on cognitive load: Eye tracking and dynamic audiovisual texts,' in *Proceedings of the 2013 Conference on Eye Tracking South Africa*, ser. ETSA '13, New York, NY, USA: Association for Computing Machinery, 29th Aug. 2013, pp. 62–66, ISBN: 978-1-4503-2110-5. DOI: 10.1145/2509315.2509331. [Online]. Available: https://doi.org/10.1145/2509315.2509331 (visited on 22/05/2022).

[40] B. Laeng, S. Sirois and G. Gredebäck, 'Pupillometry: A window to the preconscious?' *Perspect Psychol Sci*, vol. 7, no. 1, pp. 18–27, 1st Jan. 2012, Publisher: SAGE Publications Inc, ISSN: 1745-6916. DOI: 10.1177/1745691611427305. [Online]. Available: https://doi.org/10.1177/1745691611427305 (visited on 23/05/2022).

[41] S. Pietinen, R. Bednarik, T. Glotova, V. Tenhunen and M. Tukiainen, 'A method to study visual attention aspects of collaboration: Eye-tracking pair programmers simultaneously,' presented at the Proceedings of the 2008 symposium on Eye tracking research & applications, Savannah, Georgia: Association for Computing Machinery, 2008, pp. 39–42. DOI: 10.1145/1344471.1344480. [Online]. Available: https://doi.org/10.1145/1344471.1344480.

[42] K. Holmqvist, M. Nyström and F. Mulvey, 'Eye tracker data quality: What it is and how to measure it,' in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '12, New York, NY, USA: Association for Computing Machinery, 28th Mar. 2012, pp. 45–52, ISBN: 978-1-4503-1221-9. DOI: 10.1145/2168556.2168563. [Online]. Available: https://doi.org/10.1145/2168556.2168563 (visited on 08/06/2022).

[43] D. C. Niehorster, T. H. W. Cornelissen, K. Holmqvist, I. T. C. Hooge and R. S. Hessels, 'What to expect from your remote eye-tracker when participants are unrestrained,' *Behav Res*, vol. 50, no. 1, pp. 213–227, 1st Feb. 2018, ISSN: 1554-3528. DOI: 10.3758/s13428-017-0863-0. [Online]. Available: https://doi.org/10.3758/s13428-017-0863-0 (visited on 08/06/2022).

[44] 'Your eye tracker 5,' Tobii Help Center. (), [Online]. Available: https://help.tobii.com/hc/en-us/articles/360008926158-Your-Eye-Tracker-5 (visited on 07/06/2022).

[45] B. Schneider and R. Pea, 'Real-time mutual gaze perception enhances collaborative learning and collaboration quality,' *Intern. J. Comput.-Support. Collab. Learn.*, vol. 8, no. 4, pp. 375–397, 1st Dec. 2013, ISSN: 1556-1615. DOI: 10.1007/s11412-013-9181-4. [Online]. Available: https://doi.org/10.1007/s11412-013-9181-4 (visited on 09/11/2021).

[46] S. D'Angelo and D. Gergle, 'An eye for design: Gaze visualizations for remote collaborative work,' presented at the Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal QC, Canada: Association for Computing Machinery, 2018, Paper 349. DOI: 10.1145/3173574.3173923. [Online]. Available: https://doi.org/10.1145/3173574.3173923.

[47] N. Yao, J. Brewer, S. D'Angelo, M. Horn and D. Gergle, 'Visualizing gaze information from multiple students to support remote instruction,' in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '18, New York, NY, USA: Association for Comput-

ing Machinery, 20th Apr. 2018, pp. 1–6, ISBN: 978-1-4503-5621-3. DOI: 10.1145/3170427.3188453. [Online]. Available: https://doi.org/10.1145/3170427.3188453 (visited on 30/11/2021).

[48] W. C. Hill, J. D. Hollan, D. Wroblewski and T. McCandless, 'Edit wear and read wear,' in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '92, New York, NY, USA: Association for Computing Machinery, 1st Jun. 1992, pp. 3–9, ISBN: 978-0-89791-513-7. DOI: 10.1145/142750.142751. [Online]. Available: https://doi.org/10.1145/142750.142751 (visited on 10/11/2021).

[49] B. Schneider and R. Pea, 'Using eye-tracking technology to support visual coordination in collaborative problem-solving groups,' Jun. 2013, Accepted: 2019-05-22T09:50:59Z Publisher: International Society of the Learning Sciences. DOI: 10.22318/cscl2013.1.406. [Online]. Available: https://repository.isls.org//handle/1/1935 (visited on 09/11/2021).

[50] T. Van Gog, '11 the signaling (or cueing) principle in multimedia learning,' *The Cambridge handbook of multimedia learning*, p. 263, 2014, Publisher: Cambridge University Press, ISSN: 1107035201.

[51] R. E. Mayer, 'Unique contributions of eye-tracking research to the study of learning with graphics,' *Learning and instruction*, vol. 20, no. 2, pp. 167–171, 2010, Publisher: Elsevier, ISSN: 0959-4752.

[52] B. B. De Koning, H. K. Tabbers, R. M. Rikers and F. Paas, 'Attention guidance in learning from a complex animation: Seeing is understanding?' *Learning and instruction*, vol. 20, no. 2, pp. 111–122, 2010, Publisher: Elsevier, ISSN: 0959-4752.

[53] J.-M. Boucheix and R. K. Lowe, 'An eye tracking comparison of external pointing cues and internal continuous cues in learning with complex animations,' *Learning and instruction*, vol. 20, no. 2, pp. 123–135, 2010, Publisher: Elsevier, ISSN: 0959-4752.

[54] K. Kiili and H. Ketamo, 'Eye-tracking in educational game design,' *The Danish School of Education*, pp. 160–167, 2010.

[55] L. A. Williams and R. R. Kessler, 'All i really need to know about pair programming i learned in kindergarten,' *Commun. ACM*, vol. 43, no. 5, pp. 108–114, 1st May 2000, ISSN: 0001-0782. DOI: 10.1145/332833.332848. [Online]. Available: https://doi.org/10.1145/332833.332848 (visited on 25/11/2021).

[56] P. Baheti, L. Williams, E. Gehringer and D. Stotts, 'Exploring pair programming in distributed object-oriented team projects,' presented at the Educator's Workshop, OOPSLA, Citeseer, 2002, pp. 4–8.

[57] M.-A. Nüssli, 'Dual eye-tracking methods for the study of remote collaborative problem solving,' EPFL, 2011.

[58] A. Begel and N. Nagappan, 'Pair programming: What's in it for me?' In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ser. ESEM '08, New York, NY, USA: Association for Computing Machinery, 9th Oct. 2008, pp. 120–128, ISBN: 978-1-59593-971-5. DOI: 10.1145/1414004.1414026. [Online]. Available: https://doi.org/10.1145/1414004.1414026 (visited on 05/04/2022).

[59] J. E. Hannay, T. Dybå, E. Arisholm and D. I. K. Sjøberg, 'The effectiveness of pair programming: A meta-analysis,' *Information and Software Technology*, Special Section: Software Engineering for Secure Systems, vol. 51, no. 7, pp. 1110–1122, 1st Jul. 2009, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2009.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584909000123 (visited on 02/12/2021).

[60] B. J. Oates, *Researching Information Systems and Computing*. SAGE, 2006, 364 pp., ISBN: 978-1-4129-0224-3.

[61] P. Kruchten, 'The 4+1 view model of architecture,' *IEEE Software*, vol. 12, no. 6, pp. 42–50, Nov. 1995, Conference Name: IEEE Software, ISSN: 1937-4194. DOI: 10.1109/52.469759.

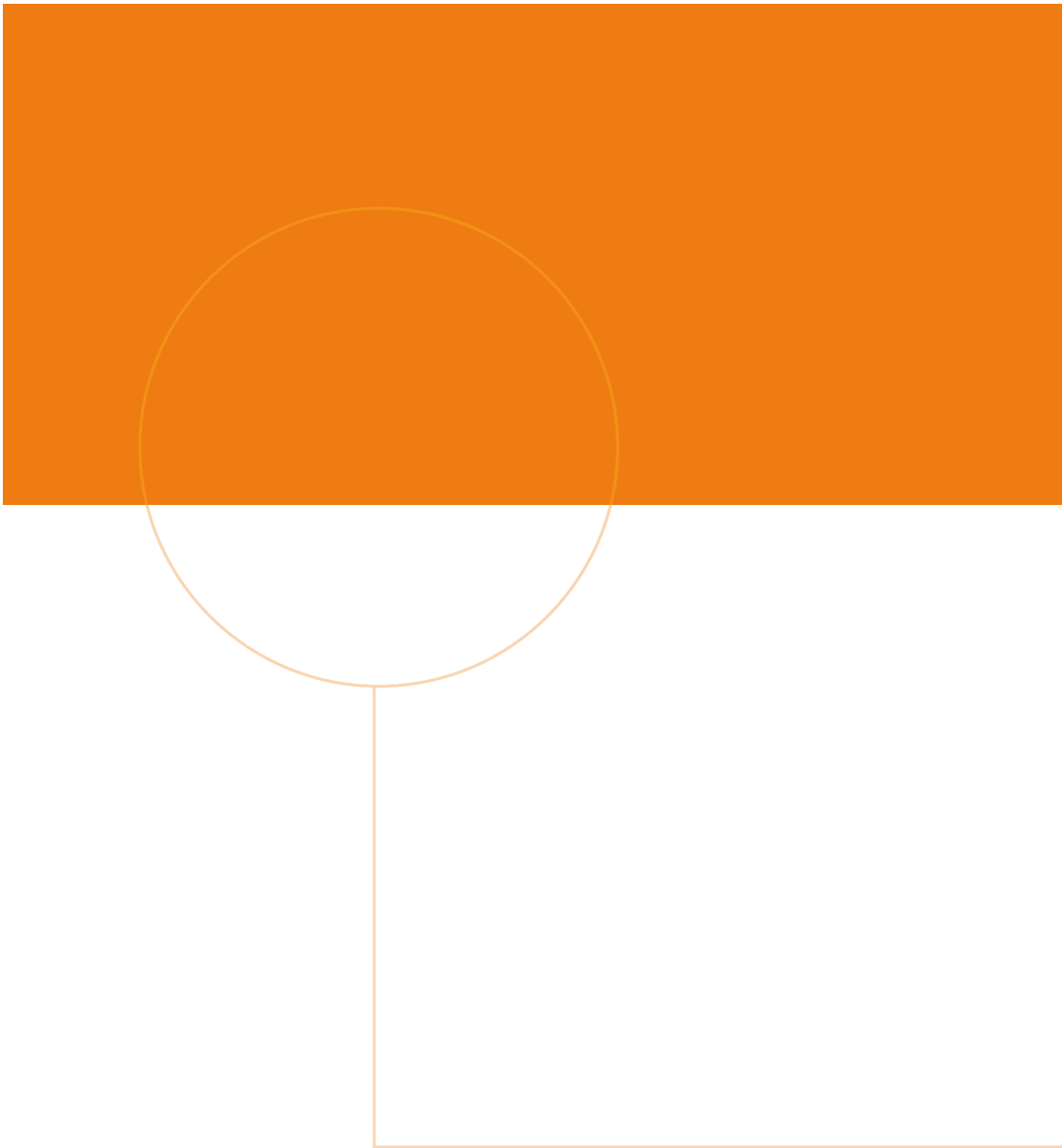
[62] C. Anderson, 'Presenting and evaluating qualitative research,' *AJPE*, vol. 74, no. 8, 1st Oct. 2010, Publisher: American Journal of Pharmaceutical Education Section: Special Articles, ISSN: 0002-9459, 1553-6467. DOI: 10.5688/aj7408141. [Online]. Available: https://www.ajpe.org/content/74/8/141 (visited on 30/05/2022).

[63] B. G. Glaser, 'The constant comparative method of qualitative analysis,' *Social Problems*, vol. 12, no. 4, pp. 436–445, 1965, Publisher: [Oxford University Press, Society for the Study of Social Problems], ISSN: 0037-7791. DOI: 10.2307/798843. [Online]. Available: https://www.jstor.org/stable/798843 (visited on 30/05/2022).

[64] J. Huang, R. White and G. Buscher, 'User see, user point: Gaze and cursor alignment in web search,' in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12, New York, NY, USA: Association for Computing Machinery, 2012, pp. 1341–1350, ISBN: 978-1-4503-1015-

4. DOI: 10.1145/2207676.2208591. [Online]. Available: https://doi.org/10.1145/2207676.2208591 (visited on 06/06/2022).

[65] P. Burnard, P. Gill, K. Stewart, E. Treasure and B. Chadwick, 'Analysing and presenting qualitative data,' *Br Dent J*, vol. 204, no. 8, pp. 429–432, Apr. 2008, Number: 8 Publisher: Nature Publishing Group, ISSN: 1476-5373. DOI: 10.1038/sj.bdj.2008.292. [Online]. Available: https://www.nature.com/articles/sj.bdj.2008.292 (visited on 01/06/2022).



NTNU

Norwegian University of
Science and Technology