Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3723010.3723018

RESEARCH-ARTICLE

A Cookbook for Eye Tracking in Software Engineering

LISA GRABINGER, Regensburg University of Applied Sciences, Regensburg, Bayern, Germany

NASER AL MADI, Colby College, Waterville, ME, United States

ROMAN BEDNARIK, University of Eastern Finland, Kuopio, IS, Finland

TERESA BUSJAHN, Berlin University of Applied Sciences, Berlin, Germany

FABIAN ENGL, Regensburg University of Applied Sciences, Regensburg, Bayern, Germany

TIMUR EZER, Regensburg University of Applied Sciences, Regensburg, Bayern, Germany

View all

Open Access Support provided by:

Abertay University

Polytechnique Montral

Colby College

University of Malaya

University of Nebraska–Lincoln

Regensburg University of Applied Sciences

View all

PDF Download
3723010.3723018.pdf
01 February 2026
Total Citations: 0
Total Downloads: 730

Published: 02 June 2025

Citation in BibTeX format

ECSEE 2025: European Conference on Software Engineering Education
June 2 - 4, 2025
Seeon, Germany

https://doi.org/10.1145/3723010.3723018
ISBN: 9798400712821

A Cookbook for Eye Tracking in Software Engineering

Lisa Grabinger
OTH Regensburg
Regensburg, Germany
lisa.grabinger@oth-regensburg.de

Naser Al Madi
Colby College
Waterville, ME, USA
nsalmadi@colby.edu

Roman Bednarik
University of Eastern Finland
Joensuu, Finland
roman.bednarik@uef.fi

Teresa Busjahn
HTW Berlin
Berlin, Germany
teresa.busjahn@htw-berlin.de

Fabian Engl
OTH Regensburg
Regensburg, Germany
fabian.engl@oth-regensburg.de

Timur Ezer
OTH Regensburg
Regensburg, Germany
timur.ezer@oth-regensburg.de

Hans Gruber
University of Regensburg
Regensburg, Germany
Hans.Gruber@ur.de

Florian Hauser
OTH Regensburg
Regensburg, Germany
florian.hauser@oth-regensburg.de

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

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

Kang-il Park
University of Nebraska-Lincoln
Lincoln, NE, USA
kangil.park@huskers.unl.edu

Bonita Sharif
University of Nebraska-Lincoln
Lincoln, NE, USA
bsharif@unl.edu

Zohreh Sharafi
Polytechnique Montréal
Montréal, Canada
zohreh.sharafi@polymtl.ca

Lynsay Shepherd
Abertay University
Dundee, United Kingdom
lynsay.shepherd@abertay.ac.uk

Jürgen Mottok
OTH Regensburg
Regensburg, Germany
juergen.mottok@oth-regensburg.de

Abstract

Eye tracking technology offers valuable insights into how developers and users interact with software artifacts, tools, and interfaces. However, conducting empirical eye tracking research comes with a number of challenges. To assist researchers and students new to the field, this article provides a concise summary of the background as well as the key considerations. As an additional resource, we present a detailed checklist along with its application. Note that both the outline and the checklist are specifically tailored to, but not limited to, the context of software engineering research.

CCS Concepts

• **Software and its engineering**; • **General and reference** → *Reference works*;

Keywords

reference guide, empirical research, checklist

This work is licensed under a Creative Commons Attribution International 4.0 License.

ECSEE 2025, Seeon, Germany
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1282-1/25/06
https://doi.org/10.1145/3723010.3723018

ACM Reference Format:
Lisa Grabinger, Naser Al Madi, Roman Bednarik, Teresa Busjahn, Fabian Engl, Timur Ezer, Hans Gruber, Florian Hauser, Jonathan I. Maletic, Unaizah Obaidellah, Kang-il Park, Bonita Sharif, Zohreh Sharafi, Lynsay Shepherd, and Jürgen Mottok. 2025. A Cookbook for Eye Tracking in Software Engineering. In *ECSEE 2025: European Conference on Software Engineering Education (ECSEE 2025), June 02–04, 2025, Seeon, Germany*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3723010.3723018

1 Introduction

Using eye tracking technology is becoming increasingly important in the software engineering community [53, 90, 113]. It allows researchers to empirically collect data on what developers actually perceive while performing software engineering activities (e.g., reviewing code). Conducting eye tracking studies inherently involves human subjects and, as such, requires a specific scientific process and rigor to produce valid and meaningful results. Unfortunately, expertise in conducting human subject experiments is typically outside the purview of many software engineering researchers. In addition, using eye tracking to study human behavior presents several challenges that need to be clarified.

The goal of this paper is to articulate a foundational framework for conducting eye tracking experiments within the context of software engineering. We provide a self-contained resource outlining the basic theoretical and methodological issues of eye tracking and how it is applied to software engineering. Additionally, we define a step-by-step process and checklist (see Table 2) for designing and conducting eye tracking studies. Also included are some of the

pitfalls and general problems that the authors have experienced in conducting a very large number of studies over the years. Our hope is that this cookbook will assist researchers (and students), new to the area, save time and produce sound results.

Note that there are already similar methodological works focused on either eye tracking (e.g. [32, 74]) or software engineering (e.g., [132], [106]). However, to the best of our knowledge, there is no work that is concise enough for beginners and takes into account the special circumstances of software engineering research.

The paper is organized as follows. Section 2 covers the basics of eye tracking, while section 3 gives a brief overview of its use in software engineering research. The following sections focus on how to conduct eye tracking studies in software engineering – they outline a step-by-step process (see section 4), accompanying aspects (see section 5), and general limitations of the research method (see section 6). Section 7 concludes with hands-on recommendations.

2 Theoretical Background

This section explains the basic terms in eye tracking research – the biological and technical background as well as the typical interpretation of the collected data.

2.1 Functionality

We now discuss the basic biological and technical aspects necessary to understand how eye tracking data extraction works.

Visual Field The eye is a sensory organ, an extension of the brain, that can actively be directed towards stimuli. Light enters through the cornea, which is located in the front of the eye. The retina, lining the back of the eye, hosts photoreceptors: *rods*, which are light-sensitive but do not detect color, and *cones*, which process color and have high spatial resolution. The density of those photoreceptors varies between different regions of the retina. The fovea centralis, a small patch near the center of the retina densely packed with cones, is the zone of highest visual acuity. It allows one to sharply perceive an area of about the size of a thumbnail when the arm is outstretched. Outside the fovea, visual acuity decreases. This allows for efficient processing: central vision for detail and peripheral vision for broader awareness. [21, 40, 63]

In reading, the perceptual span – the functional visual field – is asymmetric and influenced by a number of factors such as reading direction, text difficulty, and reading skill [103].

Pupil-Cornea-Reflex Several methods exist for measuring eye movements. The most commonly used is infrared pupil-corneal reflection tracking. This method employs infrared light directed at the eye, creating four reflections from different parts of the eye. The most prominent reflection is from the cornea. The point of regard is determined from the position of this corneal reflection relative to the center of the pupil. As the eye moves, their spatial relationship changes, allowing for the calculation of gaze location. [40, 63]

Stimuli The term stimulus refers to the visual artifact presented to the participants to elicit responses for research purposes. In the field of software engineering, stimuli typically include artifacts such as source code, UML diagrams, and documentation (see section 3).

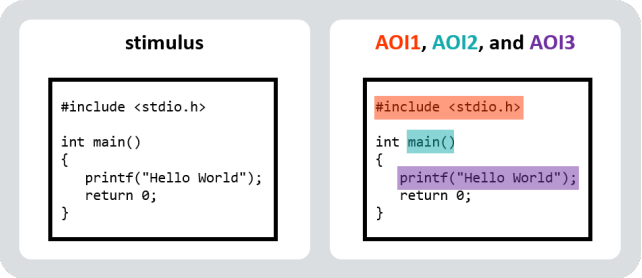

Figure 1: Three exemplary AOIs for a sample stimulus with a C code snippet.

Calibration Calibration means recording the participants looking at known points on a stimulus so the system can learn to map the participants' gaze accurately. Typically, it includes a series of predefined points such as 5, 9, or 13, which cover the area where stimuli are presented. Using the recorded pupil and corneal reflection positions at these points, any gaze location on the stimulus can be estimated. Various factors, such as variations in eyeball physiology, visual aids, and lighting conditions, affect the calculations for determining gaze direction, and accuracy is highest near the calibration targets [40, 63].

The decision to accept or repeat the calibration is usually made by the researcher on the basis of the data obtained – summarized in numbers such as accuracy or visualized by points and error vectors.

2.2 Interpretation

Eye movement analysis reveals how people perceive and process visual information, offering insights into their interaction with the environment [40]. This section details the theoretical constructs used for interpretation.

Areas of Interest Stimuli are usually segmented into *areas of interest* (AOIs) that encompass pertinent components or features. AOIs help to focus on specific elements during analysis. They are often distinct and meaningful segments of the stimulus, which vary in size and shape. Alternatively, AOIs can be created by overlaying the stimulus with a grid and using the grid cells as AOIs, regardless of their content [63]. When using source code as a stimulus, AOIs can include individual code elements (i.e., tokens), entire code lines, or larger segments of code. An example of AOIs on a source code snippet can be found in Figure 1.

Movements Typically, eye tracking data is reduced to two movements that play complementary roles in visual and cognitive processes, i.e., fixations and saccades. *Fixations* occur when the eyes remain stationary, focusing on a specific point to allow the brain to process details. In contrast, *saccades* enable the eyes to quickly scan the environment, rapidly jumping from one point to another [73].

To identify fixations and saccades from the raw eye tracking data, various algorithms are used, with *dispersion-based* and *velocity-based* methods being the most widely recognized [109]. Dispersion-based algorithms, such as I-DT, cluster gaze points within a spatial threshold over a minimum duration to detect fixations. These methods are simple and effective for static stimuli but may face chal-

lenges with noise or dynamic environments. Alternatively, velocity-based algorithms, like I-VT, identify fixations when gaze velocity falls below a certain threshold. Well-suited for dynamic settings and real-time analysis, they require accurate velocity measurements and are sensitive to noise.

Metrics Historically, eye tracking data has been categorized into four main groups [69, 112], either calculated for the entire stimulus or for certain AOIs. *First-order measures* consist of raw, unprocessed data from the eye tracker, such as gaze coordinates, timestamps, and pupil size measurements; they form the foundation for all higher-level measures but require further processing to derive meaningful insights. *Second-order measures* are the identified eye movement events, such as fixations and saccades; they are essential for understanding visual attention and behavior. *Third-order measures* emerge by aggregating fixation and saccade, as counts or durations; this provides insights into attention distribution and engagement. Examples include total fixation count, average saccade duration, and overall dwell time. For an exhaustive list refer to [112]. *Fourth-order measures* cover the sequence and patterns of eye movements, such as the length of the scan path, the transitions between points of interest, and the viewing order; they provide a high-level understanding of visual exploration strategies and cognitive processing.

3 Research Areas

Eye tracking offers numerous benefits in different research areas in software engineering by revealing how developers and users interact with software artifacts, tools, and interfaces. To illustrate this variety, we detail a number of relevant areas in Table 1.

For program comprehension, eye tracking helps to uncover how developers navigate and understand source code by tracking their gaze, identifying areas of confusion, and providing insights to improve code readability and structure. In *user interface* (UI) and *user experience* (UX) research, there are four distinct research categories [23]: engineering psychology, user research, design research, and design evaluation. A key area of investigation is understanding how users interact with software interfaces and tools, such as *integrated development environments* (IDEs) and debugging tools. In the context of software visualization or modeling, eye tracking assesses how users engage with visual representations like UML diagrams or data flow graphs, identifying areas of confusion, and guiding improvements to make these tools more effective. During code reviews for safe and secure programming, it identifies how reviewers focus on specific code areas to detect errors and vulnerabilities, improving code review tools and practices to ensure higher software quality. In education and training, it targets how students engage with programming tasks, highlighting areas where additional learning support is needed and helping to design adaptive training programs. Finally, in requirements engineering, eye tracking provides insights into how stakeholders read and interpret documents, helping to eliminate ambiguity, improve communication, and ensure a common understanding of project goals.

4 Overview of the Research Process

There are four phases to an eye tracking study in software engineering, namely, planning, conducting, analyzing, and reporting (see Figure 2). What needs to be considered and reported during these phases is detailed below. Note that the explanations are not organized according to the four phases, but rather according to the general structure of reporting this information within a paper.

4.1 Design

Research Questions Research questions represent the basis for a scientific study, formulated to guide the investigation of specific phenomena. They align research objectives with the current state of knowledge and methodological approaches. A well-crafted research question precisely defines the scope of the inquiry and ensures its alignment with the intended research outcomes. *Exploratory* studies often address open-ended "W-questions" to uncover patterns, while *explanatory* research may focus on specific causal relationships, aiming at testing specific hypotheses. [24, 38, 40, 79, 81]

Hypotheses Hypotheses are structured assumptions proposed within empirical research to explain relationships between variables. They are derived from established theories or well-supported empirical evidence and are designed to be tested through experimentation. Note that hypotheses always come in pairs. The *null hypothesis* H_0 proposes the absence of an effect or relationship, while the *alternative hypothesis* H_1 asserts a specific, testable relationship. The latter are considered scientific if they have the following properties [24, 38, 40, 79, 81]:

- *Generality:* ... aim for broad applicability.
- *Falsifiability:* ... can be refuted by experimentation.
- *Internal consistency:* ... are free of contradictions.
- *Operationalizability:* ... define measurable variables.
- *Traceability:* ... are logically derived and properly justified.
- *Phrasing:* ... are often expressed in conditional forms, such as "if-then" or "as-so" statements

Research Design Essentially, two different research designs, namely *within-subjects* and *between-subjects* design, are used to test hypotheses and answer research questions. Using a within-subjects design, the same participants experience all experimental conditions, reducing variability due to individual differences. However, this design may introduce carryover effects; these effects are mostly reduced by applying randomization approaches in the study design. Between-subjects designs assign participants to different conditions, ensuring independence between groups but requiring larger sample sizes to achieve statistical power.

The choice between these designs depends on the research goals, the feasibility of randomization, and the nature of the dependent variables. It also directly impacts how independent variables are manipulated and dependent variables are measured, shaping the study's operational framework and analysis. Independent variables are the manipulated or categorized factors in a study, designed to test their impact on the outcomes – the dependent variables. These in turn explicitly represent the measured effects or responses influenced by the independent variables. The operational definition of these variables is crucial for ensuring validity and reliability in research. Proper identification and control of confounding variables are necessary to isolate causal relationships and enhance the inter-

Table 1: Research Areas of Eyetracking in Software Engineering

Area	Aspect	Characteristics
Program Comprehension	Goal	Understanding how developers read, understand, and navigate source code
	Application	• Identifying code elements that capture attention • Investigating how developers locate bugs or understand logic in unfamiliar code • Measuring cognitive load during comprehension tasks
	Artifacts	Source code in C, C++, Java, Python, Stack Overflow, GitHub pages, Bug reports...
	References	[1, 2, 9, 10, 12, 16, 22, 26–28, 58, 64, 75, 76, 84, 89, 90, 93, 94, 96, 97, 99, 101, 108, 113, 115, 116]
UI and UX Design	Goal	Understanding how users interact with software interfaces and tools
	Application	• Differentiating strategies of experts and novices (i.e., engineering psychology research) • Investigating user interaction and preferences (i.e., user research) • Evaluating the handling of UI elements (i.e., design research) • Evaluating a specific UI (i.e, design evaluation)
	Artifacts	UIs, IDEs, modeling software, (penetration-)testing tools, ...
	References	[29, 37, 49, 133, 134, 143]
Visualizations and Models	Goal	Understanding how developers perceive and interact with software visualizations
	Application	• Investigating the readability and effectiveness of software diagrams • Evaluating the impact of visualizations on comprehension and decision-making
	Artifacts	UML, SysML v2, flowcharts, ...
	References	[50–52, 68, 102, 122, 137, 140]
Code Review	Goal	Understanding the cognitive processes involved in manual code review
	Application	• Investigating how reviewers prioritize sections of the code • Differentiating gaze patterns of novice and expert reviewers • Identifying bottlenecks or inefficiencies in the review process
	Artifacts	Source code in C, C++, Java, Python, ...
	References	[1, 7, 13, 17, 26–28, 58, 59, 64, 90, 97, 98, 113–115, 125, 128, 130]
Education and Training	Goal	Enhancing teaching methods for programming and software engineering
	Application	Evaluating the effectiveness of educational materials or platforms
	Artifacts	Learning management systems and materials
	References	[15, 60, 70, 71, 85, 86, 91, 107, 112, 123, 136]
Requirements Engineering	Goal	Understanding how stakeholders interact with requirement documents or prototypes
	Application	Evaluating requirements, elicitaion techniques, or traceablity links
	Artifacts	Documented requirements, traceability links
	References	[7, 8, 61, 68, 110, 119, 131]

pretability of results. Figure 3 summarizes the influence of hypotheses on selected aspects of the research process. [24, 38, 40, 79, 81]

4.2 Sample

Participants are typically classified based on their programming expertise, including novices (e.g., undergraduate computer science students with limited coding experience) and experts (e.g., professional software developers or graduate students with extensive programming knowledge) [26, 121]. Recruitment methods often involve convenience sampling through academic institutions for novice participants via mailing lists or posters, or industry partnerships and online forums for experts [72, 100]. Eligibility criteria may include prior programming experience, familiarity with specific programming languages, and task-related expertise to ensure alignment with the study objectives [45].

Highly controlled laboratory experiments, such as eye tracking experiments, often rely on a small sample of college students as participants. Therefore, it is important to report demographics such as the age, gender, education level, programming experience, and whether the participant has normal or corrected vision (e.g., contacts or glasses). This information is often reported in aggregation so as not to reveal the identity of a specific participant.

4.3 Instruments

Stimuli Various stimuli can be utilized to investigate how programmers read, analyze, understand, and build software engineering artifacts. They can be broadly categorized as *static* (i.e., content that remains unchanged in real-time, such as images or non-animated visualizations) or *dynamic* (i.e., content that involves animations or real-time interactions) [33].

Examples of static stimuli are non-editable source code snippets (short or long) that are shown in the form of images (.jpeg or .png) in eye tracking software such as *Tobii Pro Lab*. Several factors worth considering when designing static stimuli are image size and its resolution, content layout, screen size, scrolling functionality, and a fixed viewport matched to the dimensions of the screen or window in which the image is displayed. Other formats of static stimuli include static visualizations (e.g., UML diagrams, flowcharts, design pattern layouts, pseudocode) [11, 26, 91] and dynamic visualizations (e.g., animations or step-by-step execution) [34, 117, 122].

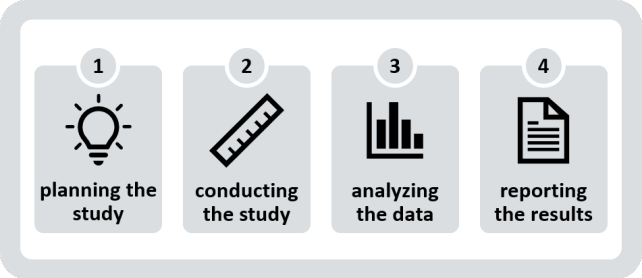

Figure 2: The four stages to complete when performing an eye tracking study.

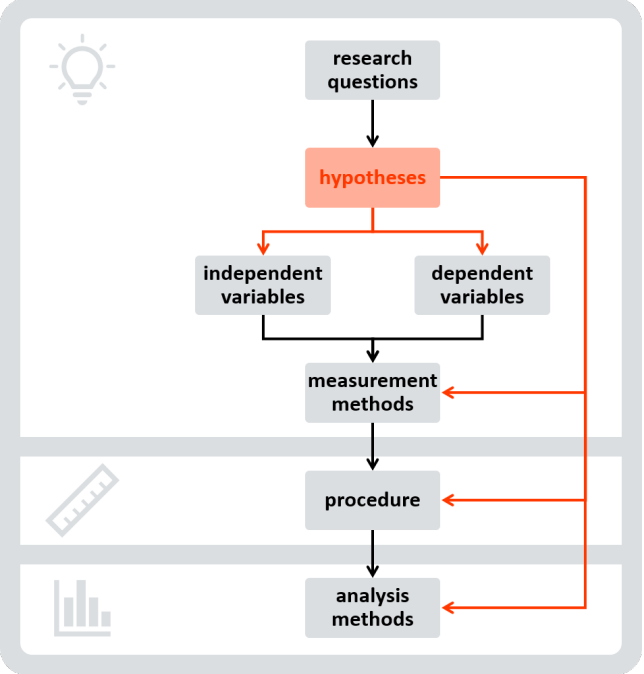

Figure 3: The influence of hypotheses on other aspects of the research process, depicted in red.

Dynamic stimuli can be further subdivided into linear and non-linear types. *Linear dynamic* stimuli refer to animated content presented uniformly to all participants, such as pre-recorded videos. These stimuli do not allow participants to alter the flow of events during the task, requiring all participants to complete the task within the same time constraints. *Non-linear dynamic* stimuli, on the other hand, involve real-time interactivity, where the content varies based on participants' individual choices during tasks. These tasks often use web-based interfaces that allow unbounded stimulus presentation and interaction possibilities. Such settings enable researchers to study, for example, developer behavior in realistic coding environments by incorporating multiple editable stimulus types, including source code files, web browsers, and coding analysis tools such as IDEs [93, 96, 118, 128]. Note that for stimuli with source code, the choice between static and dynamic stimuli defines the possible length of the code snippet – and hence leads to a further trade-off: While long code snippets may induce fatigue, short snippets may lack real-world relevance.

Tasks Each stimulus must be accompanied by a task. When formulating the task, it is important to remember that the nature of the task can change the visual attention outcome such as the pattern of eye gaze behavior. The different cognitive processes (e.g., comprehension, recall, or problem-solving) will influence strategies [135]. Additionally, the level of experience and skills of the participants may also influence their approach to the tasks.

Triangulation In eye tracking studies, *triangulation* or the *multimodal* approach refers to using multiple methods, data sources, or measures to gain a more comprehensive understanding of the participants' cognitive processes. This enhances the validity of data analysis by enabling the correlation of findings across diverse perspectives.

For example, *galvanic skin response* (GSR) captures arousal and stress levels by measuring the electrical activity of the skin, while changes in the *electroencephalogram* (EEG) spectrum reveal overall arousal, alertness [77], and the pleasantness of emotional stimuli [105]. Self-reported data on survey questionnaires such as *NASA-TLX* [3, 5] or *Self-Assessment Manikin* (SAM) [3, 47] can give insights on the participants' cognitive load, perceived difficulty of tasks, or emotional state in response to a stimulus.

Implementing these data sources provides a holistic explanation of participants' behaviors. What they do is evidenced by the eye tracking movements, while the underlying causes of those behaviors (e.g., emotional or cognitive states) can be inferred from self-reports and physiological data. With that, triangulation helps researchers disentangle ambiguities arising from single-modality analyses, offering deeper insights into the multifaceted nature of programming tasks.

However, attention should be given to data integration technicalities (e.g., aligning time-stamped data across modalities), careful interpretation of conflicting findings arising from discrepancies between data sources, and the increased complexity of resources and expertise to support the multimodal data collection and analysis.

Apparatus Currently available eye tracking solutions suitable for research purposes typically use infrared pupil-corneal reflection (see section 2.1). They can broadly be classified as head-mounted or remote. Head-mounted devices are worn by the participant and typically enable capturing eye movements along with the surrounding environment. Remote devices, which attach to the computer display, are less obtrusive [40, 63]. Regardless of the type, variations in accuracy, sampling rate, robustness, and type of data usage influence the ease of use, required researchers' ability to write analysis scripts, and, consequently, the price of the equipment. In addition to the type of device, the researcher needs to decide on a software environment to control the eye tracker with. There are many open-source options to choose from, such as *OGAMA* [129], *iTrace*[1] [57, 120], or *iTrace-DejaVu* [141, 142]. Vendor-available software such as *Tobii Pro Lab* may also be used.

[1] https://www.i-trace.org/

It is important to detail the eye tracking set up used in a study. In addition to reporting the name of the specific device (e.g., EyeLink1000), the manufacturer is also reported (e.g., SR Research Ltd.). Moreover, two performance characteristics are often reported: spatial accuracy and temporal frequency. Spatial accuracy for eye trackers is often measured in degrees of visual angle and is described as the difference between the true fixation position and the position detected by the eye tracker. Temporal accuracy, on the other hand, describes the sampling frequency of the eye tracker, and it is measured in samples per second. The sampling frequency of webcam eye trackers tends to range between 30 to 60 samples per second, while research eye trackers often take up to 1200 samples per second (i.e. 1200 Hz). However, most eye trackers can be used at different frame rates. The researcher needs to determine which frame rate is ideal for the analysis they want to conduct. A common setting is 300 Hz. There are also other aspects necessary for replication, such as screen size, refresh rate, resolution, the distance between the participant and the screen and whether a chin-rest is used or not, or lighting conditions.

4.4 Procedure

For the participants, a study typically begins with a pre-study screening, including questionnaires to assess their demographic and technical backgrounds, followed by informed consent procedures adhering to ethical research guidelines [122, 128].

If any questionnaires are collected before, during, or after the experiment, it is important to report the details of the questions and the method of collection (i.e., electronic or on paper). In addition, any experimental grouping, randomization, or counterbalancing should be reported; this includes randomizing stimuli order to avoid order effects.

4.5 Analysis

The analysis phase of eye tracking studies varies depending on previous steps, e.g., based on the research objectives or the setup used for data collection. However, there are always two key steps: the processing of raw data and the actual analysis of final data.

Data Processing Handling raw eye tracking data is challenging – primarily due to its sheer volume (e.g., 300 measurements per second for 300 Hz). To manage this, researchers typically perform event detection, i.e., determining which data points belong to fixations or saccades, marking the transition from first-order to second-order measures. Additionally, the data often requires cleaning, such as correcting for drift, which can be done before or after event detection or both. In a final step, the data can be further processed into third-order or fourth-order measures and synchronized with other data sources such as questionnaires.

Data Analysis Once the data has been pre-processed, we can draw insights from it using qualitative and quantitative analysis methods. For one, we can simply visualize the eye tracking data superimposed on the stimulus, either as a video replay or accumulated over time into a single frame. The most common visualization types thereby are *heat maps* and *gaze plots* [53, p. 11]; the former color codes the intensity of fixations in terms of frequency or duration, while the latter shows the sequence and duration of individual fixations and their connecting saccades [53, p. 11][113, p. 30]. Both are depicted as examples in Figure 4. While on-stimulus visualizations are great for exploring eye tracking data, they are rarely sufficient for addressing hypotheses. This is usually done using statistics, both descriptive and inferential. For the selection and calculation of statistical methods, for beginners we recommend the tool *eyenalyzer* [55] or the corresponding literature (e.g., [48, 49, 53, 54]).

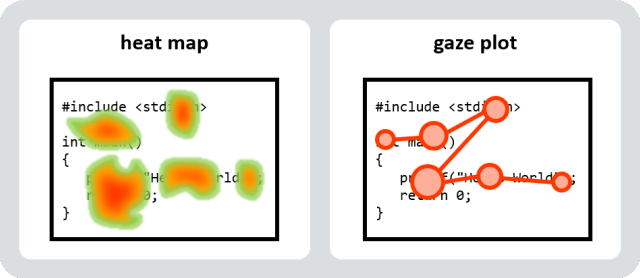

Figure 4: Exemplary on-stimulus visualizations for the sample stimulus from Figure 1.

Note that some eye tracking software environments come with their own processing chain, e.g., the *iTrace-Toolkit* [19] and *iTrace-Visualize* [18, 20]. Additionally, there are also publicly available post-processing scripts for specific datasets, e.g., [83] for the so-called *EMIP dataset* [14].

5 Underpinning Research Considerations

There are some aspects to consider within various phases of an eye tracking study. They are outlined in the following.

Ethical Considerations When conducting experimental studies, it is important that the researchers gain ethical approval for the work from the Institutional Ethics Committee, or *Institutional Review Board* (IRB), though individual countries may take different approaches. Participants must give informed consent [127] to participate in the study and should be told what they are expected to do in the experiment (whilst avoiding introducing bias), approximately how long it will take, how they can withdraw from the study if they feel uncomfortable, and how their data will be handled (in line with legislation such as GDPR in Europe [46]). They should also have the opportunity to ask questions. Afterwards, participants should be debriefed about the study and informed where they can learn more information.

Depending on the eye tracking technology used, some participants may not be able to be tracked, which may be related to medical conditions or physical characteristics they possess; participants should be reassured that this is not their fault. Eye tracking data can reveal a wealth of information about an individual, including their physical and mental health, skills and abilities, level of tiredness, and drug use [78]; therefore, privacy implications of gaze data should be considered [56].

Different countries vary in how they review research involving human participants, and therefore it is important to report ethical approvals (e.g. institutional reviews) and the ethical implications of the work and its contributions.

Data Management Before an eye tracking study begins, it is recommended that the researchers develop a *data management plan* (DMP) [35, 36] which outlines what data will be collected, how it will be stored, the length of time it will be stored, and who will have access to it. Data should be stored in a secure location such as a dedicated research server and be anonymized where possible. Where there is the need to link data to a participant, their unique identifier should be included in a password-protected spreadsheet that can only be accessed by the researchers.

Dissemination A paper discussing an eye tracking study is written in a way similar to other academic research articles. In general, papers contain an abstract, keywords, an introduction, related work that provides some context, a method section that draws on existing guidelines for reporting eye tracking studies [41], results, followed by a discussion and conclusions. Additional content such as stimuli or survey questions can be placed in the appendices or in a linked online repository.

An experiment with the best reproducibility would provide a complete replication package that includes all the materials used and the data collected. A replication package should include a readme-file that specifies the information that each folder and file contains. The package typically includes a folder of all the study materials providing every study task, questionnaires, etc. as provided in the experiment, and a folder of all data collected with de-identification applied. The replication package should also provide any tools the researcher created for the analysis in a separate folder specified by the replication package's readme file.

An example of a replication package for [96] can be found in [95]. In addition to a folder with the study materials and data, the replication package of Park et al. includes high-level analysis results, additional plots the authors did not include in their writing, and a zip file including all of the software projects in BlueJ they used as the study environment for participants in their highest level folder. The data folder has subfolders for each of their research questions, as they included a separate set of scripts used to analyze each research question.

Open Science Principles Research conducted should adhere to Open Science principles [42, 80], incorporating the 8 pillars of Open Science [126]. These pillars include the need for FAIR (Findable, Accessible, Interoperable, and Reusable) data in research studies and the importance of research integrity.

6 Limitations in Eye Tracking

Although the technology has advanced in recent years, there still exist important limitations that a practitioner needs to be wary of. These concern both the theoretical underpinnings of eye tracking research as well as practical limitations related to environment setup and computational processing of eye data.

Dynamics One major limitation of eye tracking is dealing with stimuli that change, i.e., dynamic stimuli such as when a participant scrolls on a page while reading. Apart from iTrace [57], vendor software is not able to accurately track gaze on AOIs during dynamic changes. One method used is to cache the entire page in a browser to account for scrolling. However, this still needs to have the entire page loaded a priori. Another problem is editing the stimuli as you are reading it. This happens, for example, when one wants to edit code. Tracking edits is a non-trivial problem and needs further investigation (see for example [44]).

Attention Any interpretation of eye tracking data is based on the assumption that what is currently in visual focus is actually being cognitively processed. This is commonly referred to as the *eye-mind assumption* [73, p. 330]. However, this is not always the case: sometimes the mind wanders. To identify such cases, triangulation can be used, e.g., think-aloud.

Event Detection Eye movements made by participants during a study need to be detected and classified in order to be able to interpret the raw gaze data and calculate eye tracking metrics [43, 111]. Hereby, it is not feasible to manually classify eye movements, as this would be a highly time-consuming process and human labelers may be susceptible to personal bias [65, 139]. Therefore, eye movement classification algorithms are utilized. Classical algorithms – based on the velocity or dispersion of recorded raw data gaze points – use manually set thresholds of certain features to classify eye movements [40, ch. 13].

Those classification algorithms that do not utilize machine learning techniques have been improved continuously over the last decades to label data with high noise and variance in noise levels, leading to an increasing number of adjustable or hard-coded parameters. Therefore, in order to make the best use of the algorithms, researchers would need a certain amount of experience and knowledge, making the eye movement classification process overly complicated [138]. For these reasons, efforts have been made to utilize machine learning algorithms to overcome the need for setting parameters in the eye movement classification algorithms manually [43, 67, 124, 138, 139]. However, individual machine learning algorithms have only recently been adapted to different eye tracking systems, and not all algorithms are openly available [43].

Data Quality Important quality factors for gaze data include the spatial discrepancy between the true gaze position and the recorded one, the temporal difference between an eye movement and its measured counterpart, and the number of valid samples collected. Addressing errors is challenging because they can vary in space and time and arise from various sources, including the hardware and software used, the recording environment, and the participants [63, 88]. Eye tracking studies in software engineering are usually less affected by temporal errors, while invalid samples and spatial errors can seriously distort the analysis.

For instance, eye movement recordings are generally susceptible to fixation *drift*. This systematic deviation between the real fixation position and the recorded position is attributed to the degradation of calibration quality over time [66], or changes in lighting or movement during the experiment [30, 62]. In reading, general drift can move the fixation from one line to another or from one word to another, negatively influencing the accuracy of any results built on the data [104]. In reading source code in particular, it has been shown that the processing tools used can influence the outcome of eye tracking research [39].

In some instances, problematic data can be preemptively avoided, for example, through frequent re-calibrations. However, problematic data typically needs to be either discarded or corrected. Although manual correction is possible, it is recommended to use automatic approaches, as they are more objective and reproducible. Several algorithms have been proposed to automatically correct drift in reading experiments, and most of these algorithms rely on heuristics that make specific assumptions about reading order and direction [31]. It has been shown that reading source code is substantially different from reading natural-language text, and therefore many of the assumptions of automatic drift correction algorithms do not hold in reading source code. More specifically, reading source code has been shown to be nonlinear [26] unlike natural-language reading, where code readers make many jumps back and forth between lines.

Automatic approaches that are of interest for software engineering related stimuli include [6, 25, 82, 87, 92]. Recently, specialized algorithms for correcting drift in reading source code have been proposed [4], with substantial improvement over using generic natural language techniques. The work presents a family of open-source algorithms for automatically correcting eye tracking data over source code, along with a manually corrected gold standard dataset through a replication package. However, the overall accuracy of most algorithms remains below 70 %, indicating that correcting eye movement over code is substantially more difficult, and therefore this problem remains open.

7 Conclusions

Conducting scientifically sound studies of human subjects with eye tracking involves a large number of decisions and details to consider. Many of these concerns are discussed in the paper to guide the reader in the design and implementation of their research. The paper aims to codify the main foundational concepts for running such studies. In addition, a checklist (see Table 2) is presented to assist the researcher in developing such a study; for better understanding, we provide an exemplary application of the checklist to the study presented in [96] (see Table 3).

The authors have a wealth of experience in conducting eye tracking studies, and with that some practical advice. Here are some of our thoughts on the subject:

- Work hard on developing clear and sound research questions. Poorly developed research questions will lead to poor and unusable results. Hence, a large amount of wasted time. Try to get input from others on the research questions.
- Performing a pilot or initial study is extremely useful. Basically, after you have a solid plan of what to study run it with a small number of participants, e.g., members of the lab not directly involved in the design. This activity will often uncover issues that were not previously considered. The knowledge learned from a small pilot study can then be used to improve the overall design of the experiment and result in a more sound and usable end result. Do not be afraid to revisit your research questions and revise them if necessary.
- When conducting the actual experiment with subjects, we recommend a very detailed step-by-step script for the moderator (researcher) to use. This makes the conduct of the study very uniform from subject to subject. It is easy for steps to be forgotten or skipped if not written down in detail. A detailed script helps to ensure scientific rigor.
- Eye tracking produces a huge amount of data. Research quality devices produce at least 60 data points (i.e. gaze points) per second; with that, a 30-minute eye tracking session produces over 100K data points. You need to plan for this amount of data in terms of storage and analysis.
- Keep very good records of your design decisions and choices. Why did you make one decision versus another alternative? This will be of great benefit when writing up your results and also for validation during analysis. Keep a detailed journal along with the checklist to document the entire process.

Lastly, reach out to other researchers with experience running eye tracking studies. Cognitive psychologists have been using this technology for much longer and are excellent colleagues to work with on such projects.

Acknowledgments

We thank Melvin Abraham and Ali Al-Ramadan for their insights and constructive feedback on the paper.

The paper is supported by the 'German Federal Ministry of Education and Research' (BMBF) within the funding project HASKI (FKZ: 16DHBKI035). We also wish to acknowledge the use of **DeepL Write** and **Writefull** to assist in the language refinement of this document. The paper remains an accurate representation of the authors' underlying work and novel intellectual contributions.

References

[1] Nahla J. Abid, Jonathan I. Maletic, and Bonita Sharif. 2019. Using developer eye movements to externalize the mental model used in code summarization tasks. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications (ETRA 2019)* (Denver, CO, USA). ACM, New York, NY, USA, Article 13, 9 pages. https://doi.org/10.1145/3314111.3319834

[2] Nahla J. Abid, Bonita Sharif, Natalia Dragan, Hend Alrasheed, and Jonathan I. Maletic. 2019. Developer reading behavior while summarizing Java methods: Size andcontext matters. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE 2019)* (Montreal, QC, Canada). IEEE, New York, NY, USA, 384–395. https://doi.org/10.1109/ICSE.2019.00052

[3] Zubair Ahsan and Unaizah Obaidellah. 2023. Effect of emotion and workload on expertise in programming. *Telematics and Informatics Reports* 11, 1 (Sep. 2023), 10 pages. https://doi.org/10.1016/j.teler.2023.100095

[4] Naser Al Madi. 2024. Advancing dynamic-time warp techniques for correcting eye tracking data in reading source code. *Journal of Eye Movement Research* 17, 1 (Jan. 2024), 9 pages. https://doi.org/10.16910/jemr.17.1.4

[5] Naser Al Madi, Siyuan Peng, and Tamsin Rogers. 2022. Assessing workload perception in introductory computer science projects using NASA-TLX. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGSE 2022)* (Providence, RI, USA). ACM, New York, NY, USA, 668–674. https://doi.org/10.1145/3478431.3499406

[6] Naser Al Madi, Brett Torra, Yixin Li, and Najam Tariq. 2025. Combining automation and expertise: A semi-automated approach to correcting eye-tracking data in reading tasks. *Behavior Research Methods* 57, 2 (Feb. 2025), 72.

[7] Nasir Ali, Zohreh Sharafi, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2015. An empirical study on the importance of source code entities for requirements traceability. *Empirical Software Engineering* 20, 2 (Apr. 2015), 442–478. https://doi.org/10.1007/s10664-014-9315-y

[8] Nasir Ali, Zohreh Sharafi, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2012. An empirical study on requirements traceability using eye-tracking. In *2012 28th IEEE International Conference on Software Maintenance (ICSM 2012)* (Trento, Italy). IEEE, New York, NY, USA, 191–200. https://doi.org/10.1109/ICSM.2012.6405271

[9] Salwa D. Aljehane, Bonita Sharif, and Jonathan I. Maletic. 2021. Determining differences in reading behavior between experts and novices by investigating eye movement on source code constructs during a bug fixing task. In *ETRA '21 Short Papers: ACM Symposium on Eye Tracking Research and Applications (ETRA*

2021) (Virtual Event). ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3448018.3457424

[10] Salwa D. Aljehane, Bonita Sharif, and Jonathan I. Maletic. 2023. Studying developer eye movements to measure cognitive workload and visual effort for expertise assessment. *Proceedings of the ACM on Human-Computer Interaction* 7, ETRA (May 2023), 18 pages. https://doi.org/10.1145/3591135

[11] Magdalena Andrzejewska and Anna Stolińska. 2022. Do structured flowcharts outperform pseudocode? Evidence from eye movements. *IEEE Access* 10, 1 (Dec. 2022), 132965–132975. https://doi.org/10.1109/ACCESS.2022.3230981

[12] Aakash Bansal, Bonita Sharif, and Collin McMillan. 2023. Towards modeling human attention from eye movements for neural source code summarization. *Proceedings of the ACM on Human-Computer Interaction* 7, ETRA (May 2023), 19 pages. https://doi.org/10.1145/3591136

[13] Roman Bednarik. 2012. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human Computer Studies* 70, 2 (Feb. 2012), 143–155. https://doi.org/10.1016/j.ijhcs.2011.09.003

[14] Roman Bednarik, Teresa Busjahn, Agostino Gibaldi, Alireza Ahadi, Mária Bieliková, Martha E. Crosby, Kai Essig, Fabian Fagerholm, Ahmad Jbara, Raymond Lister, Pavel A. Orlov, James H. Paterson, Bonita Sharif, Teemu Sirkiä, Jan Stelovsky, Jozef Tvarozek, Hana Vrzakova, and Ian van der Linde. 2020. EMIP: The eye movements in programming dataset. *Science of Computer Programming* 198, 1 (Oct. 2020), 11 pages. https://doi.org/10.1016/J.SCICO.2020.102520

[15] Roman Bednarik, Carsten Schulte, Lea Budde, Birte Heinemann, and Hana Vrzakova. 2018. Eye-movement modeling rxamples in source code comprehension: A classroom study. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling 2018)* (Koli, Finland). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3279720.3279722

[16] Roman Bednarik and Markku Tukiainen. 2006. An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the ACM Symposium on Eye tracking research & applications (ETRA 2006)* (San Diego, CA, USA). ACM, New York, NY, USA, 125–132. https://doi.org/10.1145/1117309.1117356

[17] Andrew Begel and Hana Vrzakova. 2018. Eye movements in code review. In *Proceedings of the Workshop on Eye Movements in Programming (EMIP 2018)* (Warsaw, Poland). ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3216723.3216727

[18] Joshua Behler, Gino Chiudioni, Alex Ely, Julia Pangonis, Bonita Sharif, and Jonathan I. Maletic. 2023. iTrace-Visualize: Visualizing eye-tracking data for software engineeringstudies. In *2023 IEEE Working Conference on Software Visualization (VISSOFT 2023)* (Bogotá, Colombia). IEEE, New York, NY, USA, 100–104. https://doi.org/10.1109/VISSOFT60811.2023.00021

[19] Joshua Behler, Praxis Weston, Drew T. Guarnera, Bonita Sharif, and Jonathan I. Maletic. 2023. iTrace-Toolkit: A pipeline for analyzing eye-tracking data of software engineering studies. In *Proceedings of the 45th International Conference on Software Engineering (ICSE 2023)* (Melbourne, Australiar). IEEE, New York, NY, USA, 46–50. https://doi.org/10.1109/ICSE-COMPANION58688.2023.00022

[20] Joshua A. C. Behler, Giovanni Villalobos, Julia Pangonis, Bonita Sharif, and Jonathan I. Maletic. 2024. Extending iTrace-Visualize to support token-based heatmaps and region of interest scarf plots for source code. In *2024 IEEE Working Conference on Software Visualization (VISSOFT 2024)* (AZ, USA, October). IEEE, New York, NY, USA, 139–143. https://doi.org/10.1109/VISSOFT64034.2024.00027

[21] Gary Bente. 2004. Erfassung und Analyse des Blickverhaltens. In *Lehrbuch der Medienpsychologie*, Roland Mangold, Peter Vorderer, and Gary Bente (Eds.). Hogrefe, Göttingen, Germany, 297–324.

[22] Dave Binkley, Marcia Davis, Dawn Lawrie, Jonathan I. Maletic, Christopher Morrell, and Bonita Sharif. 2013. The impact of identifier style on effort and comprehension. *Empirical Software Engineering* 18, 2 (Apr. 2013), 219–276. https://doi.org/10.1007/s10664-012-9201-4

[23] Aga Bojko. 2013. *Eye tracking the user experience: A practical guide to research.* Rosenfeld Media, Brooklyn, NY, USA.

[24] Jürgen Bortz and Christof Schuster. 2010. *Statistik für Human- und Sozialwissenschaftler* (7 ed.). Springer, Berlin, Germany.

[25] Teresa Busjahn. 2021. *Empirical analysis of eye movements during code reading: Evaluation and development of methods.* Ph. D. Dissertation. Universität Paderborn, Paderborn, Germany. https://nbn-resolving.org/urn:nbn:de:hbz:466:2-38777

[26] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: Relaxing the linear order. In *2015 IEEE 23rd International Conference on Program Comprehension (ICPC 2015)* (Florence, Italy). IEEE, New York, NY, USA, 255–265. https://doi.org/10.1109/ICPC.2015.36

[27] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. 2014. What influences dwell time during source code reading? Analysis of element type and frequency as factors. In *Proceedings of the ACM Symposium on Eye tracking research & applications (ETRA 2014)* (Safety Harbor, FL, USA). ACM, New York, NY, USA, 335–338. https://doi.org/10.1145/2578153.2578211

[28] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of code reading to gain more insight in program comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling 2011)* (Koli, Finland). ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/2094131.2094133

[29] Yaqin Cao, Yi Ding, Robert W. Proctor, Vincent G. Duffy, Yu Liu, and Xuefeng Zhang. 2021. Detecting users' usage intentions for websites employing deep learning on eye-tracking data. *Information Technology and Management* 22, 4 (Dec. 2021), 281–292. https://doi.org/10.1007/s10799-021-00336-6

[30] Michael Carl. 2013. Dynamic programming for re-mapping noisy fixations in translation tasks. *Journal of Eye Movement Research* 6, 2 (Aug. 2013), 11 pages. https://doi.org/10.16910/jemr.6.2.5

[31] Jon W. Carr, Valentina N. Pescuma, Michele Furlan, Maria Ktori, and Davide Crepaldi. 2021. Algorithms for the automated correction of vertical drift in eye-tracking data. , 556-572 pages. https://doi.org/10.3758/s13428-021-01554-0

[32] Vanessa Y Cho, Xin Hui Loh, Lyndon Abbott, Nur Anisah Mohd-Isa, and Robert P Anthonappa. 2023. Reporting eye-tracking studies in DEntistry (RESIDE) checklist. *Journal of Dentistry* 129, 1 (Feb. 2023), 8 pages. https://doi.org/10.1016/j.jdent.2022.104359

[33] Daniel Kyle Davis and Feng Zhu. 2022. Analysis of software developers' coding behavior: A survey of visualization analysis techniques using eye trackers. *Computers in Human Behavior Reports* 7, 1 (Aug. 2022), 100213. https://doi.org/10.1016/j.chbr.2022.100213

[34] Benoît De Smet, Lorent Lempereur, Zohreh Sharafi, Yann-Gaël Guéhéneuc, Giuliano Antoniol, and Naji Habra. 2014. Taupe: Visualizing and analyzing eye-tracking data. *Science of Computer Programming* 79, 1 (Jan. 2014), 260–278. https://doi.org/10.1016/j.scico.2012.01.004

[35] Deutsche Forschungsgemeinschaft (DFG). 2023. DFG-Vordruck 54.01 – Hinweise zur Anonymisierung personenbezogener Daten. https://www.dfg.de/resource/blob/168312/599de0d17fe6300d445bfaf9dabacbc9/54-01-de-data.pdf. Accessed: Accessed: 6th February 2025.

[36] Digital Curation Centre. 2024. DMPonline: Data Management Planning Tool. https://dmponline.dcc.ac.uk/. Accessed: 6th February 2025.

[37] Nghia Dinh, Lidia Dominika Ogiela, Kiet Tran-Trung, Tuan Le-Viet, and Vinh Truong Hoang. 2024. A Comprehensive analysis of cgnitive CAPTCHAs through eye tracking. *IEEE Access* 12 (Apr. 2024), 47190–47209. https://doi.org/10.1109/ACCESS.2024.3373542

[38] Nicola Döring and Jürgen Bortz. 2016. *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5 ed.). Springer, Berlin, Germany.

[39] Timon Dörzapf, Norman Peitek, Marvin Wyrich, and Sven Apel. 2024. Data Analysis Tools Affect Outcomes of Eye-Tracking Studies. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2024)* (Barcelona, Spain). ACM, New York, NY, USA, 96–106. https://doi.org/10.1145/3674805.3686672

[40] Andrew T. Duchowski. 2017. *Eye Tracking Methodology: Theory and Practice* (3 ed.). Springer, Berlin, Germany.

[41] Matt J. Dunn, Robert G. Alexander, Onyekachukwu M Amiebenomo, Gemma Arblaster, Denize Atan, Jonathan T. Erichsen, Ulrich Ettinger, Mario E Giardini, Iain D Gilchrist, Ruth Hamilton, et al. 2023. Minimal reporting guideline for research involving eye tracking (2023 edition). *Behavior Research Methods* 56, 5 (Aug. 2023), 4351–4357. https://doi.org/10.3758/s13428-023-02187-1

[42] European Commission. 2024. Open Science: Shaping Our Digital Future. https://research-and-innovation.ec.europa.eu/strategy/strategy-research-and-innovation/our-digital-future/open-science_en. Accessed: 6th February 2025.

[43] Timur Ezer, Moritz Plössl, Lisa Grabinger, Dominik Bittner, Susanne Staufer, Vamsi K. Nadimpalli, Flemming Bugert, Florian Hauser, and Jürgen Mottok. 2024. Deep learning for eye movement classification. In *17th annual International Conference of Education, Research and Innovation (ICERI 2024)* (Seville, Spain). IATED, Valencia, Spain, 4056–4065. https://doi.org/10.21125/iceri.2024.1028

[44] Sarah Fakhoury, Devjeet Roy, Harry Pines, Tyler Cleveland, Cole S. Peterson, Venera Arnaoudova, Bonita Sharif, and Jonathan I. Maletic. 2021. Gazel: Supporting source code edits in eye-tracking studies. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE 2021)* (Madrid, Spain). IEEE, New York, NY, USA, 69–72. https://doi.org/10.1109/ICSE-COMPANION52605.2021.00038

[45] Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)* (Hyderabad, India). ACM, New York, NY, USA, 402–413. https://doi.org/10.1145/2568225.2568266

[46] GDPR-Info. 2024. General Data Protection Regulation. https://gdpr-info.eu/. Accessed: 6th February 2025.

[47] Daniela Girardi, Nicole Novielli, Davide Fucci, and Filippo Lanubile. 2020. Recognizing developers' emotions while programming. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE 2020)* (Seoul, South Korea). ACM, New York, NY, USA, 666–677. https://doi.org/10.1145/3377811.3380374

[48] Lisa Grabinger, Timur Ezer, Florian Hauser, and Jürgen Mottok. 2024. The impact of eyenalyzer. In *Proceedings of the 17th annual International Conference of Education, Research and Innovation (ICERI 2024)*. IATED, Valencia, Spain, 695–701. https://doi.org/10.21125/iceri.2024.0271 Seville, Spain.

[49] Lisa Grabinger, Timur Ezer, Florian Hauser, and Jürgen Mottok. 2025. The usability of eyenalyzer. In *Proceedings of the 6th European Conference on Software Engineering Education (ECSEE 2025)* (Seeon, Germany). ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3723010.3723011

[50] Lisa Grabinger, Florian Hauser, and Jürgen Mottok. 2022. Accessing the presentation of causal graphs and an application of gestalt principles with eye tracking. In *Proceedings of the 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2022)* (Honolulu, HI, USA). IEEE, New York, NY, USA, 1278–1285. https://doi.org/10.1109/saner53432.2022.00153

[51] Lisa Grabinger, Florian Hauser, and Juergen Mottok. 2023. Evaluating graph-based modeling languages. In *Proceedings of the 5th European Conference on Software Engineering Education (ECSEE 2023)* (Seeon, Germany). ACM, New York, NY, USA, 120–129. https://doi.org/10.1145/3593663.3593664

[52] Lisa Grabinger, Florian Hauser, and Jürgen Mottok. 2024. On the perception of graph layouts. *Journal of Software: Evolution and Process* 36, 5 (May 2024), 18 pages. https://doi.org/10.1002/smr.2599

[53] Lisa Grabinger, Florian Hauser, Christian Wolff, and Jürgen Mottok. 2024. On eye tracking in software engineering. *SN Computer Science* 5, 6 (Aug. 2024), 20 pages. https://doi.org/10.1007/s42979-024-03045-3

[54] Lisa Grabinger and Jürgen Mottok. 2024. On selecting hypothesis tests for group differences. In *Proceedings of the 17th annual International Conference of Education, Research and Innovation (ICERI 2024)*. IATED, Valencia, Spain, 702–712. https://doi.org/10.21125/iceri.2024.0272 Seville, Spain.

[55] Lisa Grabinger and Jürgen Mottok. 2024. Statistical Analysis of Eye Movement Data for Beginners. In *Proceedings of the Proceedings of Mensch und Computer 2024 (MuC 2024)* (Karlsruhe, Germany). ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/3670653.3670678

[56] Céline Gressel, Rebekah Overdorf, Inken Hagenstedt, Murat Karaboga, Helmut Lurtz, Michael Raschke, and Andreas Bulling. 2023. Privacy-aware eye tracking: Challenges and future directions. *IEEE Pervasive Computing* 22, 1 (Mar. 2023), 95–102. https://doi.org/10.1109/MPRV.2022.3228660

[57] Drew T. Guarnera, Corey A. Bryant, Ashwin Mishra, Jonathan I. Maletic, and Bonita Sharif. 2018. iTrace: Eye tracking infrastructure for development environments. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA 2018)* (Warsaw, Poland). ACM, New York, NY, USA, 1–3. https://doi.org/10.1145/3204493.3208343

[58] Florian Hauser. 2024. *Visuelle Expertise bei Code Reviews*. Ph. D. Dissertation. University of Regensburg, Regensburg, Germany.

[59] Florian Hauser, Lisa Grabinger, Timur Ezer, Jürgen Horst Mottok, and Hans Gruber. 2024. Analyzing and interpreting eye movements in C++: Using holistic models of image perception. In *Proceedings of the 2024 Symposium on Eye Tracking Research and Applications (ETRA 2024)* (Glasgow, United Kingdom). ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649902.3655093

[60] Florian Hauser, Lisa Grabinger, Jürgen Mottok, Sabrina Jahn, and Vamsi Krishna Nadimpalli. 2023. The expert's vView: Eye movement modeling examples in software engineering education. In *Proceedings of the 5th European Conference on Software Engineering Education (ECSEE 2023)* (Seeon, Germany). ACM, New York, NY, USA, 148–152. https://doi.org/10.1145/3593663.3593683

[61] Florian Hauser, Rebecca Reuter, Andreas Gegenfurtner, Hans Gruber Gruber, and Jürgen Mottok. 2019. Eye movements in software modelling - What do they tell us about heuristics?. In *Proceedings of the International Conference of Education, Research and Innovation* (Seville, Spain). IATED, Valencia, Spain, 6064–6070. https://doi.org/10.21125/iceri.2019.1469

[62] Frouke Hermens. 2015. Dummy eye measurements of microsaccades: Testing the influence of system noise and head movements on microsaccade detection in a popular video-based eye tracker. *Journal of Eye Movement Research* 8, 1 (Dec. 2015), 17 pages. https://doi.org/10.16910/jemr.8.1.1

[63] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost van de Weijer. 2011. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. Oxford University Press, Oxford, England.

[64] Alexander Homann, Lisa Grabinger, Florian Hauser, and Jürgen Mottok. 2023. An eye tracking study on MISRA C coding guidelines. In *Proceedings of the 5th European Conference on Software Engineering Education (ECSEE 2023)* (Seeon, Germany). ACM, New York, NY, USA, 130–137. https://doi.org/10.1145/3593663.3593671

[65] Ignace T. C. Hooge, Diederick C. Niehorster, Marcus Nyström, Richard Andersson, and Roy S. Hessels. 2018. Is human classification by experienced untrained observers a gold standard in fixation detection? *Behavior Research Methods* 50, 5 (Oct. 2018), 1864–1881. https://doi.org/10.3758/s13428-017-0955-x

[66] Jörg Hoormann, Stephanie Jainta, and Wolfgang Jaschinski. 2008. The effect of calibration errors on the accuracy of the eye movement recordings. *Journal of Eye Movement Research* 1, 2 (Aug. 2008), 7 pages. https://doi.org/10.16910/jemr.1.2.3

[67] Sabrina Hoppe and Andreas Bulling. 2016. End-to-end eye movement detection using convolutional neural networks. https://doi.org/10.48550/arXiv.1609.02452

[68] Ivonne Hutzler, Florian Hauser, Rebecca Reuter, Jürgen Mottok, and Hans Gruber. 2018. Will the noun/verb analysis be used to generate class diagrams? An eye tracking study. In *Proceedings of the 11th annual International Conference of Education, Research and Innovation (ICERI 2018)* (Seville, Spain). IATED, Valencia, Spain, 505–514. https://doi.org/10.21125/iceri.2018.1103

[69] Robert J.K. Jacob and Keith S. Karn. 2003. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, Jukka Hyönä, Ralf Radach, and Heiner Deubel (Eds.). Elsevier, Amsterdam, Netherlands, 573–605. https://doi.org/10.1016/B978-044451020-4/50031-1

[70] Halszka Jarodzka, Kenneth Holmqvist, and Hans Gruber. 2017. Eye tracking in educational science: Theoretical frameworks and research agendas. *Journal of Eye Movement Research* 10, 1 (Jan. 2017), 18 pages. https://doi.org/10.16910/jemr.10.1.3

[71] Halszka Jarodzka, Irene Skuballa, and Hans Gruber. 2021. Eye-tracking in educational practice: Investigating visual perception underlying teaching and learning in the classroom. https://doi.org/10.1007/s10648-020-09565-7

[72] Sarah Jessup, Sasha M Willis, Gene Alarcon, and Michael Lee. 2021. Using eye-tracking data to compare differences in code comprehension and code perceptions between expert and novice programmers. In *Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS 2021)*. HICSS, Honolulu, HI, USA, 114–124. https://doi.org/10.24251/HICSS.2021.013

[73] Marcel Just and Patricia Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological Review* 87 (July 1980), 329–354. Issue 4. https://doi.org/10.1037/0033-295x.87.4.329

[74] Enkelejda Kasneci, Hong Gao, Suleyman Ozdel, Virmarie Maquiling, Enkeleda Thaqi, Carrie Lau, Yao Rong, Gjergji Kasneci, and Efe Bozkir. 2024. Introduction to eye tracking: A hands-on tutorial for students and Practitioners. https://doi.org/10.48550/arXiv.2404.15435

[75] Katja Kevic, Braden Walters, Timothy Shaffer, Bonita Sharif, David C. Shepherd, and Thomas Fritz. 2017. Eye gaze and interaction contexts for change tasks - Observations and potential. *Journal of Systems and Software* 128, 1 (June 2017), 252–266. https://doi.org/10.1016/J.JSS.2016.03.030

[76] Katja Kevic, Braden M. Walters, Timothy R. Shaffer, Bonita Sharif, David C. Shepherd, and Thomas Fritz. 2015. Tracing software developers' eyes and interactions for change tasks. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)* (Bergamo, Italy). ACM, New York, NY, USA, 202–213. https://doi.org/10.1145/2786805.2786864

[77] Arthur F. Kramer. 2020. Physiological metrics of mental workload: A review of recent progress. In *Multiple task performance*, Diane Damos (Ed.). CRC Press, Boca Raton, FL, USA, 279–328.

[78] Jacob Leon Kröger, Otto Hans-Martin Lutz, and Florian Müller. 2020. What does your gaze reveal about you? On the privacy implications of eye tracking. In *15th IFIP International Summer School on Privacy and Identity Management (Privacy and Identity 2020)* (Maribor, Slovenia). Springer, Cham, Switzerland, 226–241. https://doi.org/10.1007/978-3-030-42504-3_15

[79] Siegfried Lamnek and Claudia Krell. 2016. *Qualitative Sozialforschung* (6 ed.). Beltz, Weinheim, Germany.

[80] League of European Research Universities (LERU). 2018. Open Science and its Role in Universities: A Roadmap for Cultural Change. https://www.leru.org/files/LERU-AP24-Open-Science-full-paper.pdf. Accessed: 6th February 2025.

[81] Gustav A. Lienert and Ulrich Raatz. 1998. *Testaufbau und Testanalyse* (6 ed.). Beltz, Weinheim, Germany.

[82] Sebastia Lohmeier. 2015. Experimental Evaluation and Modelling of the Comprehension of Indirect Anaphors in a Programming Language. http://www.monochromata.de/master_thesis/ma1.3.pdf. Accessed: 6th February 2025.

[83] Naser Al Madi, Drew T. Guarnera, Bonita Sharif, and Jonathan I. Maletic. 2021. EMIP toolkit: A Python library for customized post-processing of the eye movements in programming dataset. In *ETRA '21 Short Papers: ACM Symposium on Eye Tracking Research and Applications (ETRA 2021)* (Virtual Event). ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3448018.3457425

[84] Naser Al Madi, Cole S. Peterson, Bonita Sharif, and Jonathan I. Maletic. 2021. From novice to expert: Analysis of token level rffects in a longitudinal eye tracking study. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC 2021)* (Madrid, Spain). IEEE, New York, NY, USA, 172–183. https://doi.org/10.1109/ICPC52881.2021.00025

[85] Niloofar Mansoor, Cole S Peterson, Michael D. Dodd, and Bonita Sharif. 2024. Assessing the effect of programming language and task type on eye movements of computer science students. *ACM Transactions on Computing Education* 24, 1 (Mar. 2024), 38 pages. https://doi.org/10.1145/3632530

[86] Jürgen Horst Mottok, Florian Hauser, Lisa Grabinger, Timur Ezer, and Fabian Engl. 2024. An educational perspective on eye tracking in engineering sciences. In *Proceedings of the 2024 Symposium on Eye Tracking Research and Applications (ETRA 2024)* (Glasgow, United Kingdom). ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649902.3653945

[87] Marc-Antoine Nüssli. 2011. *Dual Eye-Tracking Methods for the Study of Remote Collaborative Problem Solving.* Ph. D. Dissertation. École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. http://infoscience.epfl.ch/record/169609

[88] Marcus Nyström, Richard Andersson, Kenneth Holmqvist, and Joost van de Weijer. 2012. The influence of calibration method and eye physiology on eyetracking data quality. *Behavior Research Methods* 45 (Mar. 2012), 272–288. https://doi.org/10.3758/s13428-012-0247-4

[89] Unaizah Obaidellah and Mohammed Al Haek. 2018. Evaluating gender difference on algorithmic problems using eye-tracker. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA 2018)* (Warsaw, Poland). ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3204493.3204537

[90] Unaizah Obaidellah, Mohammed Al Haek, and Peter C.-H. Cheng. 2018. A survey on the usage of eye-tracking in computer programming. *Comput. Surveys* 51, 1 (Jan. 2018), 58 pages. https://doi.org/10.1145/3145904

[91] Unaizah Obaidellah, Tanja Blascheck, Drew T. Guarnera, and Jonathan Maletic. 2020. A fine-grained assessment on novice programmers' gaze patterns on pseudocode problems. In *ACM Symposium on Eye Tracking Research and Applications (ETRA 2020 Short Papers)* (Stuttgart, Germany). ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379156.3391982

[92] Christopher Palmer and Bonita Sharif. 2016. Towards Automating Fixation Correction for Source Code. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications (Etra 2016)* (Charleston, SC, USA). ACM, New York, NY, USA, 65–68. https://doi.org/10.1145/2857491.2857544

[93] Kang-il Park, Jack Johnson, Cole S. Peterson, Nishitha Yedla, Isaac Baysinger, Jairo Aponte, and Bonita Sharif. 2024. An eye tracking study assessing source code readability rules for program comprehension. *Empirical Software Engineering* 29, 6 (Nov. 2024), 160. https://doi.org/10.1007/S10664-024-10532-X

[94] Kang-il Park and Bonita Sharif. 2021. Assessing perceived sentiment in pull requests with emoji: Evidence from tools and developer eye movements. In *2021 IEEE/ACM Sixth International Workshop on Emotion Awareness in Software Engineering (SEmotion 2021)* (Madrid, Spain). IEEE, New York, NY, USA, 6 pages. https://doi.org/10.1109/SEMOTION52567.2021.00009

[95] Kang-il Park and Bonita Sharif. 2024. An eye tracking study assessing the impact of background styling in code editors on novice programmers' code understanding - Project Results. https://osf.io/3uprw/

[96] Kang-il Park, Pierre Weill-Tessier, Neil C. C. Brown, Bonita Sharif, Nikolaj Jensen, and Michael Kölling. 2023. An eye tracking study assessing the impact of background styling in code editors on novice programmers' code understanding. In *Proceedings of the 2023 ACM Conference on International Computing Education Research (CIER 2023)* (Chicago, IL, USA). ACM, New York, NY, USA, 444–463. https://doi.org/10.1145/3568813.3600133

[97] Patrick Peachock, Nicholas Iovino, and Bonita Sharif. 2017. Investigating eye movements in natural language and C++ source code - A replication experiment. In *Proceedings of the Augmented Cognition (AC 2017)* (Vancouver, BC, Canada). Springer, Cham, Germany, 206–218. https://doi.org/10.1007/978-3-319-58628-1_17

[98] Cole S. Peterson, Nahla J. Abid, Corey A. Bryant, Jonathan I. Maletic, and Bonita Sharif. 2019. Factors influencing dwell time during source code reading: a large-scale replication experiment. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications (ETRA 2019)* (Denver, CO, USA). ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3314111.3319833

[99] Cole S. Peterson, Kang-il Park, Isaac Baysinger, and Bonita Sharif. 2021. An eye tracking perspective on how developers rate source code readability rules. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW 2021)* (Melbourne, Australia). IEEE, New York, NY, USA, 138–139. https://doi.org/10.1109/ASEW52652.2021.00037

[100] Cole S. Peterson, Jonathan A. Saddler, Tanja Blascheck, and Bonita Sharif. 2019. Visually Analyzing Students' Gaze on C++ Code Snippets. In *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP 2019)* (Montreal, QC, Canada). IEEE, New York, NY, USA, 18–25. https://doi.org/10.1109/EMIP.2019.00011

[101] Cole S. Peterson, Jonathan A. Saddler, Natalie M. Halavick, and Bonita Sharif. 2019. A gaze-based exploratory study on the information seeking behavior of developers on stack overflow. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA 2019)* (Glasgow, United Kingdom). ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3290607.3312801

[102] Gerardo Cepeda Porras and Yann-Gaël Guéhéneuc. 2010. An empirical study on the efficiency of different design pattern representations in UML class diagrams. *Empirical Software Engineering* 15, 5 (Oct. 2010), 493–522. https://doi.org/10.1007/s10664-009-9125-9

[103] Keith Rayner, Barbara J. Juhasz, and Alexander Pollatsek. 2005. Eye Movements During Reading. In *The Science of Reading: A Handbook*, Margaret J. Snowling and Charles Hulme (Eds.). John Wiley & Sons, Hoboken, NJ, USA, 79–97. https://doi.org/10.1002/9780470757642.ch5

[104] Erik D. Reichle and Denis Drieghe. 2015. Using EZ Reader to examine the consequences of fixation-location measurement error. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 41, 1 (Jan. 2015), 262. https://doi.org/10.1037/a0037090

[105] Boris Reuderink, Christian Mühl, and Mannes Poel. 2013. Valence, arousal and dominance in the EEG during game play. *International Journal of Autonomous and Adaptive Communications Systems* 6, 1 (Dec. 2013), 45–62. https://doi.org/10.1504/IJAACS.2013.050691

[106] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (Apr. 2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8

[107] Jonathan A. Saddler, Cole S. Peterson, Patrick Peachock, and Bonita Sharif. 2019. Reading behavior and comprehension of C++ source code - A classroom study. In *Augmented Cognition: 13th International Conference (AC 2019)* (Orlando, FL, USA) *(Lecture Notes in Computer Science, Vol. 11580)*. Springer, Cham, Switzerland, 597–616. https://doi.org/10.1007/978-3-030-22419-6_43

[108] Jonathan A. Saddler, Cole S. Peterson, Sanjana Sama, Shruthi Nagaraj, Olga Baysal, Latifa Guerrouj, and Bonita Sharif. 2020. Studying developer reading behavior on stack overflow during API summarization tasks. In *27th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2020)* (London, ON, Canada). IEEE, New York, NY, USA, 195–205. https://doi.org/10.1109/SANER48275.2020.9054848

[109] Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications (ETRA 2000)* (Palm Beach Gardens, FL, USA). ACM, New York, NY, USA, 71–78. https://doi.org/10.1145/355017.355028

[110] Zohreh Sharafi, Alessandro Marchetto, Angelo Susi, Giuliano Antoniol, and Yann-Gaël Guéhéneuc. 2013. An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In *2013 21st International Conference on Program Comprehension (ICPC 2013)* (San Francisco, CA, USA). IEEE, New York, NY, USA, 33–42. https://doi.org/10.1109/ICPC.2013.6613831

[111] Zohreh Sharafi, Timothy Shaffer, Bonita Sharif, and Yann-Gaël Guéhéneuc. 2015. Eye-tracking metrics in software engineering. In *2015 Asia-Pacific Software Engineering Conference (APSEC 2015)* (Delhi, India). IEEE, New York, NY, USA, 96–103. https://doi.org/10.1109/APSEC.2015.53

[112] Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. 2020. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering* 25, 5 (Sep. 2020), 3128–3174. https://doi.org/10.1007/s10664-020-09829-4

[113] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. 2015. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology* 67, 7 (Nov. 2015), 79–107. https://doi.org/10.1016/j.infsof.2015.06.008

[114] Bonita Sharif, Michael Falcone, and Jonathan I. Maletic. 2012. An eye-tracking study on the role of scan time in finding source code defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA 2012)* (Santa Barbara, CA, USA). ACM, New York, NY, USA, 381. https://doi.org/10.1145/2168556.2168642

[115] Bonita Sharif, Grace Jetty, Jairo Aponte, and Esteban Parra. 2013. An empirical study assessing the effect of seeit 3D on comprehension. In *Proceedings of the 1st IEEE Working Conference on Software Visualization (VISSOFT 2013)* (Eindhoven, Netherlands). IEEE, New York, NY, USA, 10 pages. https://doi.org/10.1109/VISSOFT.2013.6650519

[116] Bonita Sharif and Jonathan I. Maletic. 2010. An eye tracking study on camelCase and under_score identifier styles. In *2010 IEEE 18th International Conference on Program Comprehension (ICPC 2010)* (Braga, Portugal). IEEE, New York, NY, USA, 196–205. https://doi.org/10.1109/ICPC.2010.41

[117] Bonita Sharif and Jonathan I Maletic. 2010. An eye tracking study on the effects of layout in understanding the role of design patterns. In *2010 IEEE International Conference on Software Maintenance (ICSM 2010)* (Timioara, Romania). IEEE, New York, NY, USA, 10 pages. https://doi.org/10.1109/ICSM.2010.5609582

[118] Bonita Sharif and Jonathan I Maletic. 2016. iTrace: Overcoming the Limitations of Short Code Examples in Eye Tracking Experiments. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME 2016)* (Raleigh, NC, USA). IEEE, New York, NY, USA, 647. https://doi.org/10.1109/ICSME.2016.61

[119] Bonita Sharif, John Meinken, Timothy Shaffer, and Huzefa H. Kagdi. 2017. Eye movements in software traceability link recovery. *Empirical Software Engineering* 22, 3 (June 2017), 1063–1102. https://doi.org/10.1007/S10664-016-9486-9

[120] Bonita Sharif, Timothy Shaffer, Jenna Wise, and Jonathan Maletic. 2016. Tracking developers' eyes in the IDE. *IEEE Software* 33, 3 (May 2016), 105–108. https://doi.org/10.1109/MS.2016.84

[121] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)* (Hyderabad, India). ACM, New York, NY, USA, 378–389. https://doi.org/10.1145/2568225.2568252

[122] Zéphyrin Soh, Zohreh Sharafi, Bertrand Van den Plas, Gerardo Cepeda Porras, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2012. Professional status and

expertise for UML class diagram comprehension: An empirical study. In *2012 20th IEEE International Conference on Program Comprehension (ICPC 2012)* (Passau, Germany). IEEE, New York, NY, USA, 163–172. https://doi.org/10.1109/icpc.2012.6240484

[123] Theresa Stark. 2021. *Learning from Eye Movement Modelling Examples: Effects on Performance and Visual Behaviour University of Regensburg Faculty of Human Sciences Department of Educational Science.* Master's thesis. University of Regensburg, Regensburg, Germany.

[124] Mikhail Startsev, Ioannis Agtzidis, and Michael Dorr. 2019. 1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits. *Behavior Research Methods* 51, 2 (Apr. 2019), 556–572. https://doi.org/10.3758/s13428-018-1144-2

[125] Rachel Turner, Michael Falcone, Bonita Sharif, and Alina Lazar. 2014. An eye-tracking study assessing the comprehension of C++ and Python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA 2014)* (Safety Harbor, FL, USA). ACM, New York, NY, USA, 231–234. https://doi.org/10.1145/2578153.2578218

[126] UCL Library Services. 2024. The 8 Pillars of Open Science. https://www.ucl.ac.uk/library/open-science-research-support/open-science/8-pillars-open-science. Accessed: 6th February 2025.

[127] UK Department for Education. 2024. Gaining Informed Consent: Ethics and Safeguarding Guidance. https://user-research.education.gov.uk/guidance/ethics-and-safeguarding/gaining-informed-consent. Accessed: 6th February 2025.

[128] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. 2006. Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA 2006)* (San Diego, CA, USA). ACM, New York, NY, USA, 133–140. https://doi.org/10.1145/1117309.1117357

[129] Adrian Voßkühler, Volkhard Nordmeier, Lars Kuchinke, and Arthur Jacobs. 2008. OGAMA (Open Gaze and Mouse Analyzer): Open-source software designed to analyze eye and mouse movements in slideshow study designs. *Behavior Research Methods* 40, 4 (Nov. 2008), 1150–1162. https://doi.org/10.3758/BRM.40.4.1150

[130] Hana Vrzakova, Andrew Begel, Lauri Mehtätalo, and Roman Bednarik. 2020. Affect recognition in code review: An in-situ biometric study of reviewer's affect. *Journal of Systems and Software* 159, 1 (Jan. 2020), 14 pages. https://doi.org/10.1016/j.jss.2019.110434

[131] Braden Walters, Timothy Shaffer, Bonita Sharif, and Huzefa H. Kagdi. 2014. Capturing software traceability links from developers' eye gazes. In *Proceedings of the 22nd International Conference on Program Comprehension (ICPC 2014)* (Hyderabad, India), Chanchal K. Roy, Andrew Begel, and Leon Moonen (Eds.). ACM, New York, NY, USA, 201–204. https://doi.org/10.1145/2597008.2597795

[132] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering.* Springer, Berlin, Germany.

[133] Baixi Xing, Xinjie Song, Qimeng Chen, Lei Shi, Yanhong Pan, Kaiqi Wang, and Mengyue Tang. 2023. User-attention based product aesthetics Eevaluation with image and eye-tracking fusion data analysis. In *2023 15th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC 2023)* (Hangzhou, China). IEEE, New York, NY, USA, 84–87. https://doi.org/10.1109/IHMSC58761.2023.00028

[134] Hao Yang, Jitao Zhang, Yueran Wang, and Ruoyu Jia. 2021. Exploring relationships between design features and system usability of intelligent car human–machine interface. *Robotics and Autonomous Systems* 143, C (Sep. 2021), 13 pages. https://doi.org/10.1016/j.robot.2021.103829

[135] Alfred Yarbus. 1967. *Eye Movements and Vision.* Springer, Cham, Switzerland.

[136] Leelakrishna Yenigalla, Vinayak Sinha, Bonita Sharif, and Martha E. Crosby. 2016. How novices read source code in introductory courses on programming: An eye-tracking experiment. In *Foundations of Augmented Cognition: Neuroergonomics and Operational Neuroscience (AC 2016)* (Toronto, ON, Canada) *(Lecture Notes in Computer Science, Vol. 9744)*. Springer, Cham, Switzerland, 120–131. https://doi.org/10.1007/978-3-319-39952-2_13

[137] Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan I. Maletic. 2007. Assessing the comprehension of UML class diagrams via eye tracking. In *IEEE International Conference on Program Comprehension (ICPC 2007)* (Banff, AB, Canada). IEEE, New York, NY, USA, 113–122. https://doi.org/10.1109/ICPC.2007.10

[138] Raimondas Zemblys, Diederick C. Niehorster, and Kenneth Holmqvist. 2019. Gazenet: End-to-end eye-movement event detection with deep neural networks. *Behavior Research Methods* 51, 2 (Apr. 2019), 840–864. https://doi.org/10.3758/s13428-018-1133-5

[139] Raimondas Zemblys, Diederick C. Niehorster, Oleg Komogortsev, and Kenneth Holmqvist. 2018. Using machine learning to detect events in eye-tracking data. *Behavior Research Methods* 50, 1 (Feb. 2018), 160–181. https://doi.org/10.3758/s13428-017-0860-3

[140] Li Zhang, Jianxin Sun, Cole S. Peterson, Bonita Sharif, and Hongfeng Yu. 2019. Exploring eye tracking data on source code via dual space analysis. In *2019 Working Conference on Software Visualization (VISSOFT 2019)* (Cleveland, OH, USA). IEEE, New York, NY, USA, 67–77. https://doi.org/10.1109/VISSOFT.2019.00016

[141] Vlas Zyrianov, Drew T. Guarnera, Cole S. Peterson, Bonita Sharif, and Jonathan I. Maletic. 2020. Automated recording and semantics-aware replaying of high-speed eye tracking and interaction data to support cognitive studies of software engineering tasks. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME 2020)* (Adelaide, Australia). IEEE, New York, NY, USA, 464–475. https://doi.org/10.1109/ICSME46990.2020.00051

[142] Vlas Zyrianov, Cole S. Peterson, Drew T. Guarnera, Joshua Behler, Praxis Weston, Bonita Sharif, and Jonathan I. Maletic. 2022. Deja Vu: Semantics-aware recording and replay of high-speed eye tracking and interaction data to support cognitive studies of software engineering tasks - methodology and analyses. *Empirical Software Engineering* 27, 7 (Dec. 2022), 39 pages. https://doi.org/10.1007/S10664-022-10209-3

[143] Melih Öder, Şükrü Eraslan, and Yeliz Yeslida. 2022. Automatically classifying familiar web users from eye-tracking data: A machine learning approach. *Turkish Journal of Electrical Engineering and Computer Sciences* 30, 1 (Jan. 2022), 233–248. https://doi.org/10.3906/elk-2103-6

Table 2: Key Considerations per Step following the Process given in Figure 3

Step	Things to Consider
Planning	❒ Clearly identify your research questions ❒ Formulate hypotheses based on the research questions ❒ Decide between within- or between-subjects design ❒ Differentiate independent and dependent variables ❒ Select appropriate eye tracking metrics used for verification/falsification of hypotheses ❒ Select appropriate statistical procedures to be used ❒ Add triangulation (as needed) ❒ Prepare a task and stimulus set for the study ❒ Ensure tasks are clear, unambiguous, and achievable within a reasonable time frame ❒ Configure user interaction method (e.g., mouse, keyboard) for answering questions and time allowed ❒ Develop a written study workflow for the study moderator to follow ❒ Submit the study protocol for Institutional Review Board (IRB)/ethics approval ❒ Perform a pilot study with someone other than researchers to uncover issues in design ❒ Gather feedback from the pilot participant and refine the study design as needed ❒ Implement statistical procedures selected earlier by writing analysis scripts and run on pilot data ❒ Select appropriate visualizations and create scripts ❒ Identify any threats to validity and limitations of the study
Conducting	❒ Ensure appropriate environmental conditions (e.g., lighting, participant position, chin rest) ❒ Select appropriate recording frequency of the eye tracker and related monitor ❒ Verify that all equipment is functional before each session ❒ Instruct the participant what to expect ❒ Allow time for participants to become familiar with the equipment ❒ Calibrate the eye tracker ❒ Record contextual data (e.g., think-aloud protocols, retrospective interviews) ❒ Use secure data storage and prepare backups
Analyzing	❒ Clean up the data ❒ Determine events ❒ Compute metrics ❒ Compute (statistical) analyses ❒ Interpret eye tracking metrics in the context of your research question ❒ Validate data quality and check for any inconsistencies ❒ Compute analyses and compare eye-tracking data with other sources (mixed method) ❒ Discuss implications of results for software engineering
Reporting	❒ Document apparatus, participants, experimental procedure, settings, and ethical implications ❒ Follow a reporting guideline like [41] or similar. ❒ Use a paper template from a scientific organization (e.g., ACM or IEEE) ❒ Present key findings ❒ Identity threats and discuss mitigation strategies and implications ❒ Consider extending your research to explore new questions or refine existing ones ❒ Provide a replication package on an appropriate online platform (e.g., Zenodo, osf.io, figshare) ❒ Participate in academic conferences or workshops
Underpinning Research Considerations	❒ Network with other researchers in this field for collaboration and feedback ❒ Write an ethical approval and ask the participants for consent ❒ Use a research data management system (see FAIR principles)

Table 3: Exemplary Application of the Checklist from Table 2 for the Study from [96]

Step	Things to Consider
Planning	☑ Clearly identify your research questions RQ1: Does presentation modality affect the accuracy of users on code comprehension tasks? RQ2: Does presentation modality affect the speed of users on code comprehension tasks? RQ3: Does presentation modality affect where users look during code comprehension tasks? ☑ Formulate hypotheses based on the research questions This is an exploratory study since researchers could not hypothesize in advance, which of the three presentation modalities was better than the other for the comprehension tasks. ☑ Decide between within- or between-subjects design This was a within-subjects study where each participant was exposed to all treatments of the independent variable. A cross-over trial of three different programming interfaces (three treatments) was used. ☑ Differentiate independent and dependent variables Independent variables: Task Type (method-scope summarization (SS), code-scope summarization (CS), logical/code bug (CB), functional bug (FB)), Highlighting condition (No scope highlighting (JN), scope highlighting (JS), Stride using frames (S)). Dependent variables: Accuracy (Correct, Close to correct, Partially correct, Incorrect), Time to complete, Fixation Count per second, Fixation Duration per second, Saccade Length, Linearity Metrics [26] (Vertical Next Text, Vertical Later Text, Horizontal Later Text, Regression Rate, Line Regression, Line Coverage). ☑ Select appropriate eye tracking metrics used for verification/falsification of hypotheses Fixation Count per second, Fixation Duration per second, Saccade Length, Linearity Metrics [26] (Vertical Next Text, Vertical Later Text, Horizontal Later Text, Regression Rate, Line Regression, Line Coverage). The areas of interest for number of fixations and fixation duration include field declarations, field assignment, if/else, method comments, if, bug lines, call buggy method, and ambiguous method. ☑ Select appropriate statistical procedures to be used 2-tailed independent sample t-Test for a paired comparison between each of the pairs of the scope highlighting conditions (JN vs. JH), (JN vs. S), (JH vs. S). Effect sizes are also reported. ☑ Add triangulation (as needed) The screen was recorded to enable the verification of the participants' activity during the completion of the tasks and the evaluation of their answers' correctness and speed. ☑ Prepare a task and stimulus set for the study Twelve tasks were prepared in each of the three conditions (JH, JN, S). The code was taken from three Java systems (ImageViewer, SpaceGame, and WaveLab). Refer to the replication package [95] for a full list of tasks and code base of systems. Tutorial task was based on the Java text command-based game World of Zuul. ☑ Ensure tasks are clear, unambiguous, and achievable within a reasonable time frame The researchers iterated through the tasks by themsevles but also with people not affiliated with the research group. This helped reassess the wording for the prompt and the interview and also indicated how long the study would typically take. ☑ Configure user interaction method (e.g., mouse, keyboard) for answering questions and time allowed The researchers used the iTrace-BlueJ plugin (for the BlueJ editor) for this study. The study was done in the BlueJ editor with eye tracking enabled. The participant was allowed to interact with the code by scrolling and moving between files. For each task, the prompt was asked verbally by the moderator with the participant's verbal responses recorded for later processing. They were not given a time limit. If responses were unclear, the moderator prompted for further clarification. ☑ Develop a written study workflow for the study moderator to follow The replication package [95] has a pdf checklist file with the written workflow for the moderator to follow with sections on what to do before the participant arrives, when the participant arrives and after the participant leaves. ☑ Submit the study protocol for Institutional Review Board (IRB)/ethics approval The online form for IRB approval was completed explaining the study design, data collection, institutions involved, recruitment, and participants. All the tasks and study material in the replication package [95] were uploaded to this online system and submitted to each site's IRB for approval. ☑ Perform a pilot study with someone other than researchers to uncover issues in design Since this study was done on two sites, each site tested this with one undergraduate student to estimate how long the tasks would take and if the prompts were well defined.

Continued from Table 3

Step	Things to Consider
	☑ Gather feedback from the pilot participant and refine the study design as needed After the pilot, the researchers changed some prompts that were not very clear. ☑ Implement statistical procedures selected earlier by writing analysis scripts and run on pilot data The researchers ran the data through iTrace-Toolkit (had to be extended to support the Stride (S) treatment because it was a little different than the JH and JN treatments) to generate fixations on the areas of interest (line and token). The statistical analysis was done on the fixations. The replication package [95] has all the scripts implementing the statistical procedures. ☑ Select appropriate visualizations and create scripts Scarfplots were used to visualize transitions between areas of interest, box plots for descriptive statistics, and bar graphs for time and accuracy measurements. The replication package [95] has all the scripts. ☑ Identify any threats to validity and limitations of the study Differences in eye tracking metrics between the Stride (S) and Java (JH, JN) conditions can simply be a result of the differences in syntax of the languages themselves. There was only one participant who had experience with the Stride language.
Conducting	☑ Ensure appropriate environmental conditions (e.g., lighting, participant position, chin rest) The study was conducted in a controlled lab environment in a room with no windows. Standard ceiling-mounted lighting was present. A chin rest was not used in this study. Since this study was done at two sites, each site ensured the seating arrangement and setup were the same. Two computers were used during the study, one was for the participant and the other for the moderator. A 24-inch monitor was used for study tasks on the participant computer. The moderator's computer was used for recording data. This was done to avoid the recording software interacting with the eye tracking software. Two webcams were connected to the moderator's computer, where one webcam recorded the participant's face and the other recorded the screen from behind the participant's shoulder via a tripod. A voice recorder was used to record participant's responses as answers were given verbally. ☑ Select appropriate recording frequency of the eye tracker and related monitor The recording frequency of the eye tracker was set to 120 Hz. The monitor of the participant was set at a resolution of 1920x1080. ☑ Verify that all equipment is functional before each session Before conducting the study: Ensure eye tracker works through the Tobii Eye Tracker Manager software. Ensure webcams are functional on moderator's computer. Reserve enough memory space for the voice recorder and keep it charged. During the study: Monitor Eye Tracker manager to make sure the eye tracker is still functional. Check the iTrace-BlueJ plugin window to see whether plugin data is being recorded. There was a little number on the status bar that kept counting up if the data was being recorded as indication for the moderator. ☑ Instruct the participant what to expect Read through and go over with participant the study overview document that explains general instructions on how to position yourself for an eye tracker and what tasks will be performed. In addition, a brief presentation going over BlueJ and each of the BlueJ projects was given to each participant. The replication package has all these documents [95]. ☑ Allow time for participants to become familiar with the equipment After the overview was provided to participant, they were given a practice code summarization and bug finding task on a sample BlueJ project. This let them know what is expected as a response. ☑ Calibrate the eye tracker Calibrate eye tracker via Tobii Eye Tracker Manager and verify calibration using iTrace-Core's calibration. ☑ Record contextual data (e.g., think-aloud protocols, retrospective interviews) For each study task, participants were presented with one Java or Stride file in the BlueJ environment and given a question asking to summarize a method or find a bug in the code. The eye tracker recorded eye movement data while reading code. The voice recorder recorded the answers given by participant in addition to any thoughts voluntarily spoken out loud. ☑ Use secure data storage and prepare backups Data is stored in a locked lab computer and backed up to external hard drives locked away in cabinet. A shared OneDrive directory is also used to share data across sites.

Continued from Table 3

Step	Things to Consider
Analyzing	☑ Clean up the data After using iTrace-Toolkit to process the raw data to fixation data, the data was then consolidated into one csv file via an R script as seen in the replication package [95]. The fixation data was used by scripts which then computed metrics and analyses. ☑ Determine events The researchers ran the data through iTrace-Toolkit to generate fixations using the IDT algorithm with a duration of 10 ms and dispersion window of 125 pixels. ☑ Compute metrics The correctness and duration of task answers were marked based on video examination. An R script was written to generate eye tracking metrics (dependent variables) based on the cleaned data as seen in the replication package [95]. ☑ Compute (statistical) analyses The researchers used the generated metrics to run statistical tests between the three different conditions via a separate R script. Since they used pairwise t-tests for each metric and each pair of conditions, they corrected for multiple comparisons using Benjamini-Hochberg False Discovery Rate (FDR) procedure with a base alpha of 0.05. ☑ Interpret eye tracking metrics in the context of your research question (relevant to eye tracking) RQ3: There is a significant difference between Stride and each of the other two Java interfaces for fixation counts for all tasks. For fixation duration, there is a medium effect size between JN and JH as well as between JN and Stride. There were shorter saccade lengths across all tasks in Stride, with a large effect than JH and a medium effect with JN. Participants are shown to navigate around elements of the code less when reading code written in Stride perhaps spending more time focused around the relevant areas of the task. All participants who answered the task question correctly were reading the buggy lines toward the end of the task. ☑ Validate data quality and check for any inconsistencies An R script from the replication package [95] was written to inspect any missing or duplicate data visually. This was done by checking for duplicate fixations and whether a participant had fixations on all tasks. ☑ Compute analyses and compare eye-tracking data with other sources (mixed methods) The screen recording data was referenced against the computed eye tracking data to determine the accuracy of what was being looked at. ☑ Discuss implications of results for software engineering The results showed that when the programming language is identical, there is no notable difference in how developers looked at code when scope-based highlighting was added to an IDE. Even though they did not find any performance related differences, the eye tracking linearity metrics did show some significant differences between the conditions.
Reporting	☑ Document apparatus, participants, experimental procedure, settings, and ethical implications Hardware and software apparatus used in study were listed and described in their respective sections. The participant section listed the number of participants from each institution and the demographic information as tables. Experimental procedure and settings were specified in the study protocol section. Ethical implications were addressed by prefacing the methods section with the approval numbers from each institution's respective research ethics boards. ☑ Follow a reporting guideline like [41] or similar. This paper did not follow [41] verbatim but had all the required reporting organised in specific sections titled "Apparatus", "Hardware", "Software", "Study Protocol", "Data Collection", "Conditions and Design", "Participants", "Areas of Interest", "Task answers correctness and duration marking". The results section has a section for each RQs results. A "Discussion and Implications" section preceeded the conclusions. ☑ Use a paper template from a scientific organization (e.g., ACM or IEEE) ICER 2023 required ACM's "sigconf" template in LaTeX. ☑ Present key findings There were less overall eye movements in Stride compared to Java. However, fixations were longer on Stride code. There were few differences between the two Java conditions.

Continued from Table 3

Step	Things to Consider
	☑ Identity threats and discuss mitigation strategies and implications It is unclear whether the differences between Java and Stride were due to the presentation of code or the differences in syntax of the languages themselves. Only one participant had prior experience with the Stride language. These are addressed in the discussion section. ☑ Consider extending your research to explore new questions or refine existing ones One can further investigate why there are differences between Stride and Java specifically. Also, one can recruit participants that have prior experience with Stride. ☑ Provide a replication package on an appropriate online platform (e.g., Zenodo, osf.io, figshare) A replication package is available [95]. ☑ Participate in academic conferences or workshops Two of the authors presented their work at the ICER conference (https://icer2023.acm.org/track/icer-2023-papers/) in Chicago, August 8-10, 2023.
Underpinning Research Considerations	☑ Network with other researchers in this field for collaboration and feedback The researchers attended the conference, talked about their work with other researchers, and solicited feedback. ☑ Write an ethical approval and ask the participants for consent Before data collection for the study began, each institution sought approval from their respective research boards for the project. Once both institutions' protocols were approved, informed consent forms were provided to each participant before beginning the study. ☑ Use a research data management system (see FAIR principles) The study was designed with FAIR principles in mind. The study's replication package is detailed enough allowing other researchers access and reuse of stimuli, tasks, and scripts.