Universität Regensburg

Visuelle Expertise bei Code Reviews

Inaugural-Dissertation zur Erlangung der Doktorwürde der Fakultät für Humanwissenschaften der Universität Regensburg

Florian Hauser (M.A.)

Betreut von
Prof. Dr. Dr. h.c. Hans Gruber (Universität Regensburg)
Prof. Dr. Jürgen Mottok (OTH Regensburg)

1. April 2024

Für meinen Sohn Simon und meine Frau Ines

Zusammenfassung

Die vorliegende Dissertation hat das Ziel, visuelle Expertise bei Code Reviews mittels Eye-Tracking zu erforschen. Sie greift in ihren theoretischen Annahmen auf die Grundlagen der Expertiseforschung, sowie auf die aus der Radiologie stammenden *holistic models of image perception* zurück und wendet diese im Kontext des Software Engineerings an. Die Eigenschaften von visueller Expertise werden sowohl für die prozedurale Programmiersprache C, als auch für die objektorientierte Programmiersprache C++ in jeweils einer Studie untersucht.

Die C-Studie adressiert primär die Fähigkeit Fehler in einem Quellcode zu finden. Sie basiert auf den Forschungsdesigns früherer Arbeiten von anderen Forschern, erweitert deren Ansätze jedoch. Das Design wird mit neuen theoretischen Grundlagen, zusätzlichen Erhebungsmethoden und tiefergehenden Analysen ergänzt. Sie kann auf die Daten von 23 Versuchspersonen ($n_{(Novizen)}$=15; $n_{(Experten)}$=8) zurückgreifen, deren Augenbewgungen von einem SMI 250RED mobile Eye-Tracker erfasst worden sind. Die Ergebnisse legen nahe, dass sich erfahrungsbedingte Unterschiede zwischen den Experten und Novizen ergeben: Experten sind in der Lage mehr Fehler mit weniger visuellem Aufwand zu finden. Ebenso zeigt sich, dass die Reviews in Phasen ablaufen, die von jeweils dominanten Strategien gekennzeichnet sind.

Die C++-Studie bedient sich eines ähnlichen Designs und einer ähnlichen Methodik, ergänzt aber korrekte Beispiele als Distraktoren. Ihre Stichprobe umfasst 34 Versuchspersonen ($n_{(Novizen)}$=18; $n_{(Experten)}$=16). Die Augenbewgungen werden in dieser Studie von einem Tobii Pro Spectrum aufgezeichnet. Die Ergebnisse deuten ebenfalls an, dass sich die Experten und Novizen in ihrer Vorgehensweise unterscheiden und Erfahrung einen Einfluss auf die Durchführung des Reviews ausübt. Ebenso kann bestätigt werden, dass die Begutachtung der Quellcodes in Phasen abläuft, welche erneut jeweils von einer dominanten Strategie gekennzeichnet sind.

Insgesamt liefern die Ergebnisse der beiden Studien Belege dafür, dass sich die *holistic models of image perception* für die Analyse und Interpretation von Augenbewegungen während eines Code Reviews eignen und visuelle Expertise in dieser Domäne erklären können. Abschließend werden diese Modelle vereinheitlicht und ein Impuls dafür gegeben, wie zukünftige Studien zu dieser Thematik aussehen könnten.

Abstract

The aim of this PhD thesis is to investigate visual expertise in code reviews using eye tracking. In its theoretical assumptions, it draws on the theories of expertise research as well as on the *holistic models of image perception* originating from radiology and applies these in the context of software engineering. The properties of visual expertise are investigated for both the procedural programming language C and the object-oriented programming language C++ in one study each.

The C study primarily addresses the ability to find errors in source code. It is based on the research designs of earlier work by other researchers, but extends their approaches. The design is supplemented with new theoretical foundations, additional survey methods and in-depth analyses. It uses data from 23 subjects ($n_{(novices)}$=15; $n_{(experts)}$=8) whose eye movements are recorded by an SMI 250RED mobile eye tracker. The results suggest that there are experience-related differences between the experts and novices: Experts are able to find more errors with less visual effort. It is also apparent that the reviews are carried out in phases that are each characterized by dominant strategies.

The C++ study uses a similar design and methodology, but adds correct examples as distractors. Its sample comprises 34 subjects ($n_{(novices)}$=18; $n_{(experts)}$=16). The eye movements in this study are recorded by a Tobii Pro Spectrum. The results also indicate that the experts and novices differ in their approach and that experience has an influence on the way how the review is conducted. It can also be confirmed that the review of the source codes takes place in phases, each of which is again characterized by a dominant strategy.

Overall, the results of the two studies provide evidence that *holistic models of image perception* are suitable for analyzing and interpreting eye movements during a code review and can explain visual expertise in this domain. Finally, these models are combined and an impulse for future studies on this topic is given.

Danksagung

Ich möchte mich an dieser Stelle bei einer ganzen Reihe von Personen bedanken, die mich bei der Erstellung dieser Dissertation auf die verschiedensten Arten und Weisen unterstützt haben.

Beginnen möchte ich mit meinen Kolleginnen und Kollegen vom Laboratory for Safe and Secure Systems an der OTH Regensburg:

Den Anfang mache ich hier mit meinem ehemaligen Kollegen Markus Reis, der gleichzeitig auch der erste Masterstudent war, den ich während seines Studiums betreuen und unterstützen durfte. Egal ob für Eye-Tracking-Studien (damals noch mit Hard- und Software von SMI) Probanden, Stimuli oder spezielle Auswertungsprogramme benötigt wurden, Markus war immer gerne zur Stelle und hat viele Dinge schnell und unkompliziert gelöst. Gleichzeitig hat mich Markus so sehr mit seiner Leidenschaft für das Programmieren angesteckt, dass ich mir diese Fähigkeit so schnell wie möglich für die Datenauswertungen und die Experimentaldesigns aneignen wollte. Auch die gemeinsamen Stadionbesuche beim SSV Jahn Regensburg waren immer eine willkommene Abwechslung.

Ebenfalls bedanken möchte ich mich bei einem weiteren Masterstudenten. Diesmal geht mein Dank an Daniel Muckelbauer. Daniel hat mich bei der Erstellung von Stimuli zu meiner C++-Studie unterstützt und sich mit großem Interesse in die Datensätze der Tobii-Eye-Tracker eingearbeitet. Gerade letzteres half mir sehr dabei, mich mit den Tobii-Geräten vertraut zu machen.

Ebenso möchte ich noch Stefan Schreistetter erwähnen, der im Rahmen des EVELIN-Projekts mein letzter Masterand war und dessen Hilfe beim Aufbau und der Inbetriebnahme des Eye-Tracking-Classrooms von enormer Bedeutung waren. Seien es Admin-Tätigkeiten oder die Neuverkabelung des Labors, Stefan hatte immer gute Ideen und hat diese stets unkompliziert und entspannt umgesetzt.

Großer Dank gilt auch Timur Ezer, der sich 2023 dem LaS^3 2023 anschloss. In Bezug auf die Auswertung von Eye-Tracking-Daten ist es immer gut Mathematiker an Board zu haben und Timur war mir bei mancher Diskussion ein hervorragender Partner und hat mir zahlreiche Ideen und Impulse gegeben. Gerade die Diskussionen, wie sich Veränderungen bei den verschiedenen Eye-Tracking-Metriken mathematisch bemerkbar machen könnten werden mir immer gut im Gedächtnis bleiben. Ebenso freut es mich sehr, dass er sich in seiner Dissertation mit der Anwendung von KI im Eye-Tracking beschäftigt und die im Rahmen dieser Dissertation erhobenen Daten von ihm aktiv für Trainingszwecke in diesem Bereich genutzt werden und

sie nicht nur irgendwo auf einer Festplatte schlummern.

Mein größter Dank hinsichtlich meiner Kollegen gilt Lisa Grabinger, die mir immer mit Rat und Tat zur Seite stand und mit der ich viele (und lange) Diskussionen über unsere Promotionsthemen geführt habe. Lisa brachte mir ebenfalls viele Impulse und gerade bei der Erstellung von Tabellen und Grafiken für diese Dissertation war sie mir (und dem ganzen Team im LaS^3) eine unglaubliche Hilfe. Besonders bedanken möchte ich mich bei ihr für Ihre Unterstützung bei der Erstellung der Abbildungen zu den *holistic models of image perception*, welche im Rahmen dieser Dissertation eine zentrale Rolle spielen und von uns auch in mehreren Publikationen verwendet werden. Herzlich bedanken möchte ich mich auch für ihr Review dieser Arbeit, welches mir bei der Überarbeitung geholfen hat.

Außerhalb der OTH gab es natürlich ebenfalls viele Menschen, die mich unterstützt haben und bei denen ich mich ebenfalls bedanken möchte.

Mein besonderer Dank gilt meiner ehemaligen Kollegin (und passenderweise auch ehemaligen Nachbarin) Dr. Rebecca Reuter. Uns verbinden viele gemeinsame Jahre im LaS^3, viele Dienstreisen und so manches Hoch und Tief, welches wir in der Arbeit erleben mussten oder durften. Rebecca hat mich vor allem während des EVELIN-Projekts hinsichtlich der Informatik unterstützt und mich dazu motiviert mich mit all den Vorteilen dieser Disziplin zu beschaffen. Sie hat mich auch dazu animiert mich beim Schreiben von Word zu lösen und auf einen Editor (damals noch *Atom* mit der Sprache *Markdown*) zu setzen (letztendlich bin ich doch noch bei *LaTeX* gelandet ...).

Mein herzlicher Dank gilt Dr. Markus Nivala, der mir im Sommer 2015 die Grundlagen des Eye-Trackings und die Durchführung von eigenen Studien beigebracht hat. Aus unserer ersten gemeinsamen Datenerhebung ist auch das Paper *Developing visual expertise in software engineering: An eye-tracking-study* entstanden, bei dem ich Co-Autor bin und welches ich 2016 bei meiner ersten Dienstreise für die OTH Regensburg auf der IEEE EDUCON in Abu Dhabi vorstellen durfte. Darüber hinaus waren die Treffen mit Markus immer eine angenehme Abwechslung und meist kamen wir vom Eye-Tracking auch zu anderen (eher hobbylastigen) Themen. Hier sei noch erwähnt, dass Markus ein hervorragender Gitarrist ist!

Ebenso wäre Prof. Dr. Andreas Gegenfurtner zu erwähnen. Bei vielen Treffen und Gesprächen haben Andreas und ich immer wieder über visuelle Expertise oder Expertise im Allgemeinen gesprochen und immer wieder Ideen ausgetauscht. So gab mir auch Andreas den Tipp, mich mit den *holistic models of image perception* im Rahmen meiner Arbeit zu befassen. Dieser Rat hat all meine Experimentaldesigns und Datenauswertungen geprägt. Diese Modelle haben mich als Autor selbst nach Abschluss dieser Arbeit immer noch so sehr im Griff, dass ich sie gerne in der Zukunft weiter erforschen und für die Anforderungen des Software Engineerings anpassen möchte. Weiterhin möchte ich mich bei Andreas herzlich dafür bedanken, dass er mich im Rahmen der EARLI Konferenzen mit vielen interessanten Forscherinnen und Forschern aus aller Welt bekannt gemacht hat und mir dabei geholfen hat mich

in die SIG 17 zu integrieren, die immer eine Quelle für neue Ideen und den Austausch von Wissen ist. Auch für die Reviews meiner Paper möchte ich mich bedanken.

Nachfolgend möchte ich mich bei meinen beiden Doktorvätern bedanken. Zwei Doktorväter? Ja, die zwei Doktorväter waren insofern relevant, da es sich bei dieser Promotion um ein kooperatives Verfahren zwischen Universität und OTH Regensburg handelte. Als Doktorand ist man diesbezüglich in einer überaus angenehmen Situation: man lernt auf diese Weise beide Hochschulen kennen, kann sein Netzwerk erweitern und bekommt (wie in meinem Fall) Feedback aus verschiedenen Richtungen. Wenn man dann noch gut betreut ist (was bei mir ebenfalls zutraf), hat man als Doktorand eine sehr interessante Zeit während der Promotion.

Beginnen möchte ich bei Prof. Dr. Jürgen Mottok, der an der OTH Regensburg mein Doktorvater war und mich stets bei der Erstellung dieser Dissertation oder dem gesamten Verlauf des Promotionsverfahrens unterstützt hat. Die Zusammenarbeit mit Jürgen war mir stets eine Freude und rückblickend ist es schön zu sehen, welche Früchte sie trägt. Hier sind neben zahlreichen gemeinsamen Publikationen, Lehrveranstaltungen und Dienstreisen, dieser Dissertation und erfolgreicher Förderanträge auch der Regensburger Eye-Tracking-Classroom zu nennen, der im Rahmen der vorliegenden Arbeit eine große Rolle gespielt hat. Ich möchte mich auch noch explizit bei Jürgen dafür bedanken, dass er mir die Möglichkeit gegeben hat, nach dem EVELIN-Projekt zurück an die OTH Regensburg zu kommen und mir frühzeitig ermöglicht hat, auch an der Erstellung von Förderanträgen mitzuwirken und so einen Überblick über das komplexe politische System zu erlangen, welches Unternehmungen wie diese Dissertation überhaupt erst ermöglichen.

Mein großer Dank gilt meinem Doktorvater Prof. Dr. Dr. h. c. Hans Gruber an der Universität Regensburg. Ich habe unter Prof. Grubers Betreuung bereits meine Bachelorarbeit und meine Masterarbeit angefertigt und habe mich im Verlauf der Promotion stets über sein Feedback gefreut, welches mir immer weiterhalf. Weiterhin möchte ich mich auch bei ihm dafür bedanken, dass er mich bereits im Studium mit seinen Lehrveranstaltungen für das Thema Expertise begeistert hat, welches mich auch nach Abgabe dieser Dissertation noch verfolgen wird und mich sowohl im Privaten, als auch im Kontext meiner Arbeit interessiert.

Besonders bedanken möchte ich mich bei meinen Freunden und meiner Familie, die mich bei der Anfertigung dieser Dissertation unterstützt haben.

Hier möchte ich mit meiner ehemaligen Kommilitonin Martina Hartl beginnen. Mit Martina verbinden mich die Studienzeit, unzählige gemeinsame Projektarbeiten und eine langjährige Freundschaft. Insofern war es naheliegend sie um Feedback zu bitten. Martina war so freundlich diese Dissertation zu lesen und mir ein paar Impulse mit auf den Weg zu geben, welches ich dankend angenommen und eingebaut habe.

Großer Dank gilt auch meinen Eltern, die mich immer unterstützt haben und mich stets dazu ermuntert haben, mich selbst zu verwirklichen und neue Dinge auszuprobieren. Auch die Ideen eines Studiums und einer anschließenden Promotion

befürworteten sie und haben mir geholfen, wo immer es ging.

Ebenfalls großer Dank gilt meiner Frau Ines, die mich bei der Erstellung dieser Dissertation immer nach Kräften unterstützt hat. Für mich war es immer interessant die Gedanken meiner Frau zu dieser Doktorarbeit und meinen Ideen diesbezüglich zu hören. Da Ines einen Hintergrund in der Betriebswirtschaftslehre hat, war sie häufig weit genug von der Thematik entfernt und konnte meine Arbeit aus einem anderen Blickwinkel betrachten, der mir neue Einsichten ermöglicht hat.

Abschließend gilt mein größter Dank noch meinem Sohn Simon. Auch wenn er zum Zeitpunkt der Abgabe dieser Dissertation noch viel zu klein ist um zu wissen, was ich in den langen Nächten getan habe, als er in seiner Babywippe saß und nicht schlafen wollte, war er mir in dieser Zeit doch die größte annehmbare Hilfe. Ein Blick zur Seite genügte und ich wusste wieder, für wen und weshalb ich diese Dissertation verfasse.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitungsgedanken

"Wir können die moderne Welt ohne Software nicht aufrechterhalten", lautet eine Aussage von Ian Sommerville (2012, S.28) in seinem Standardwerk *"Software Engineering"*. Mit diesem, auf den ersten Blick eher düster anmutenden und übertrieben wirkenden Zitat will Sommerville (2012) darauf hinweisen, dass unser gesamtes modernes Alltagsleben inzwischen stark digitalisiert ist und Computer bzw. die darauf laufenden Programme in vielen Bereichen unserer Gesellschaft eine tragende Rolle spielen. Inwiefern sich Sommervilles (2012) Zitat bewahrheitet und welche Relevanz verlässlich arbeitende Software für unsere Gesellschaft hat, soll nachfolgend näher beschrieben werden.

1.1 Der gesellschaftliche Wert von Software im 21. Jahrhundert

Software lässt sich inzwischen in nahezu jedem Bereich unseres Lebens finden. Wir werden in unserem Alltagsleben beispielsweise vom Wecker unseres Smartphones geweckt, legen unseren Weg zur Arbeit mit einem voll digitalisierten (und vielleicht bald autonom fahrenden) Fahrzeug zurück, führen finanzielle Transaktionen auf dem Laptop durch und verbringen den Feierabend vor einem Smart-TV (Broy, 2006; Hoffmann, 2013). Löst man sich von dieser individuellen Ebene und betrachtet die Anwendung von Software in einem größeren Rahmen, so stellt man fest, dass diese in allen gesellschaftlichen Bereichen eine essenzielle Rolle einnimmt. So sind beispielsweise große Teile der Infrastruktur (z.B. Verkehrs- und Versorgungssystem), Fertigungs- und Verteilungsprozesse in der Industrie oder das gesamte Finanzsystem hochgradig digitalisiert. Ein Blick in die Nachrichten reicht dazu aus um zu sehen, dass das Thema der Digitalisierung auf politischer Ebene inzwischen priorisiert behandelt wird. Es zeichnete sich bereits sehr früh im Wahlkampf 2021 ab, dass mehrere Parteien den Aufbau und die Einrichtung eines eigenständigen Staatsministeriums für Digitalisierung anstreben. Bereits bestehende Förderprogramme, wie beispielsweise die *Digital-Offensive* welches das Voranschreiten der Digitalisierung auf verschiedene Weisen unterstützt, sollen sukzessiv ausgebaut und um weitere

Programme ergänzt werden (Staatsministerium für Digitalisierung, 2021; Greive & Stiens, 2021). So können beispielsweise bereits jetzt kleine und mittelständige Unternehmen im Rahmen des Programms *"Digital Bayern"* Fördergelder beantragen, die ihre Abläufe verstärkt digitalisieren wollen (Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie, 2020), Schulen werden unter dem Titel *Digitale Schule 2020* mit neuer IT-Technik ausgestattet (Bayerisches Staatsministerium für Unterricht und Kultus, 2020) und an Universitäten und Hochschulen werden entweder neue Fakultäten gegründet, die sich vorrangig mit Informatik beschäftigen (Knobloch, 2019; Universität Regensburg, 2024) oder die bestehenden Einrichtungen ergänzen ihre schwerpunktmäßige Forschung und Lehre um spezifische Themen, wie beispielsweise *Big Data* oder *Künstliche Intelligenz* (Ostbayerische Technische Hochschule Regensburg, 2020a, 2020b). Die Liste von Anwendungsbeispielen ließe sich im Folgenden noch deutlich länger fortsetzen, aber bereits diese Aufzählung verdeutlicht, welche Verbreitung und welchen Stellenwert Software in unserem modernen Leben einnimmt. Zum aktuellen Zeitpunkt, mit dem allgegenwärtigen Interesse an Technologie und den durch die Corona-Pandemie aufgedeckten Lücken in der Digitalisierung, wirkt es so, als ob dieser Trend auch in nächster Zeit kein Ende finden wird und sich in den nächsten Jahren noch deutlich intensivieren wird (Hoffmann, 2013; Sommerville, 2012). Insgesamt lässt sich festhalten, dass Software in der Gesellschaft des 21. Jahrhunderts einen sehr hohen Stellenwert einnimmt.

Bei aller Euphorie über die Möglichkeiten, die die Digitalisierung für unsere Gesellschaft birgt, sollte jedoch auch Eines beachtet werden: Software mag zwar auf Computern ausgeführt werden, entwickelt wird sie aber de facto immer noch von Menschen (Ernst, Schmidt & Beneken, 2016; Hoffmann, 2013; Sommerville, 2012). Moderne Entwicklungsprozesse laufen zwar hochgradig toolunterstützt ab (Edmundson et al., 2013), sind jedoch immer noch stark von den Entwicklerinnen und Entwicklern und deren individuellen Fähigkeiten abhängig. Gleichzeitig gilt es zu bedenken, dass es sich bei Software in allen Aspekten des Entwicklungs- und Wartungszyklus um abstrakte und komplexe Konstrukte handelt. Erfahrene Entwicklerinnen und Entwickler sind in ihrem normalen Arbeitsalltag beispielsweise häufig mit Quellcodes konfrontiert, die mehrere tausend, wenn nicht sogar Millionen Zeilen Umfang aufweisen (Broy, 2006; Cha, Taylor & Kang, 2019; Sillito, Murphy & De Volder, 2006). Der Umgang und das Arbeiten mit diesen muss von den in diesem Feld agierenden Personen erst erlernt werden. Insofern ist es nicht verwunderlich, dass auf dem Thema Qualitätssicherung bei Softwareprojekten ein besonderes Augenmerk liegt (Hoffmann, 2013; Kononenko, Baysal & Godfrey, 2016; Ludewig & Lichter, 2013).

Die Methoden, die mit diesem Teilaspekt der Softwareentwicklung bzw. dem *Software Engineering* verknüpft sind, werden in Kapitel 2 ausführlich behandelt. Dabei wird erläutert, wie genau sicher gestellt wird, dass die ausgelieferten Programme selbst in kritischen Situationen zuverlässig und sicher ihren Dienst ausführen.

1.2 Aktuelle Beispiele für Softwarefehler und deren Auswirkungen

Weshalb im Software Engineerig bzw. in der Softwareentwicklung der Qualitätssicherung eine so große Rolle eingeräumt wird, soll in den nachfolgenden Absätzen näher beschrieben werden (Ernst et al., 2016; Hoffmann, 2013; Kononenko et al., 2016; Sommerville, 2012). Dazu wird auf mehrere aktuelle Beispiele eingegangen, in denen Softwarefehler als Ursache für teils schwerwiegende Unfälle verantwortlich waren und welche wirtschaftlichen Konsequenzen aus fehlerbehafteten Programmen resultieren können.

1.2.1 Softwarefehler als Ursache von Unfällen

Wie bereits zuvor beschrieben wurde, finden wir Software in allen Bereichen unseres Lebens wieder. Ein besonders von der Digitalisierung betroffener Bereich ist die individuelle Mobilität, bzw. der gesamte Automotive-Sektor. So sorgte beispielsweise eine Reihe von tödlichen Verkehrsunfällen in den Jahren 2016, 2017 und 2018 für besonderes Aufsehen. Dies mag auf den ersten Blick eher banal anmuten, da man in den täglichen Nachrichten häufiger mit schweren Unfällen konfrontiert wird, in diesem Fall handelte es sich jedoch bei den involvierten Fahrzeugen in allen Fällen um autonom-fahrende Autos, die vom sogenannten Autopiloten gelenkt wurden. Neben der Unachtsamkeit bzw. Abgelenktheit der jeweiligen Fahrer, die sich viel zu stark auf die Assistenzsysteme ihrer Fahrzeuge verließen, wurden Softwarefehler in allen Fällen als eine der Hauptursachen identifiziert. So konnten beispielsweise eine Kamera und die darauf zugreifende Software eine helle LKW-Plane nicht vom Himmel unterscheiden und unterließ es, die Bremse zu betätigen. Da kein Hindernis erkannt wurde, kollidierte das Fahrzeug ungebremst (und leider mit fatalen Folgen für den Fahrer) mit dem vorbeifahrenden LKW (Banks, Plant & Stanton, 2018; Dikmen & Burns, 2017; Fleetwood, 2017; Hevelke & Nida-Rümelin, 2015).

Was in den Medien bis zu diesem Zeitpunkt häufig als eine unglaubliche Innovation in Bezug auf die Fahrzeugsicherheit und unter reißerischen Slogans wie *Keine Verkehrstoten mehr bis 2050* (Deutsche Gesetzliche Unvallversicherung, 2024; Förster, 2023) dargestellt wurde, wurde nun auf einer breiten Basis hinterfragt. Es wurden beispielsweise Überlegungen angestellt, wie viele Meilen autonome Fahrzeuge zurücklegen müssen, damit sie sich mit konventionellen Autos messen können (Kalra & Paddock, 2016). Gleichzeitig wurde auch eingeräumt, dass auch autonome Fahrzeuge nicht komplett sicher sind und Unfälle vermutlich nie voll und ganz vermieden werden können (Fleetwood, 2017; Hevelke & Nida-Rümelin, 2015).

Die derzeit bekanntesten Beispiele für Softwarefehler mit noch gravierenderen Folgen stellen die Flugzeugabstürze von zwei Boeing 737 Max8 in den Jahren 2018 und 2019 dar, bei denen insgesamt rund 300 Menschen zu Tode kamen. Soweit die noch laufenden Untersuchungen bereits Ergebnisse geliefert haben, ist die Ursache in beiden Fällen auf schwerwiegende Fehler in der Avionik-Software zurückzufüh-

ren. Dabei erfasste ein Außensensor den Steigewinkel der betroffenen Flugzeuge als zu steil und diagnostizierte einen bevorstehenden Strömungsabriss, gab diese falsche Information an den Bordcomputer weiter, welcher ihn letztendlich nach unten korrigierte und das Flugzeug somit zum Absturz brachte. Bei der Veränderung des Steigewinkels durch den Computer wurden auch die manuellen Eingabemöglichkeiten der Piloten reduziert oder vollständig gesperrt. Dadurch entstand im Cockpit ein tödlicher Kampf zwischen Mensch und Technik, den die Piloten nicht gewinnen konnten (Chappell & Gonzales, 2019; Johnston & Harris, 2019).

Abbildung 1.1: Trümmerteile einer abgestürzten Boeing 737 Max8 übernommen aus der Arbeit von Chappell und Gonzalez (2019)

Diese Vorfälle riefen in der IT-Community einige kritische Fragen hervor. So stellte Broy beispielsweise bereits im Jahr 2006 fest, dass ein normales Auto zu dieser Zeit über eine Software verfügte, die aus rund 10.000.000 Zeilen von Code besteht. Broy (2006) ging davon aus, dass sich dieser Wert von Generation zu Generation mit dem Faktor zehn multipliziert und immer komplexer wird. Ähnliche Annahmen können auch in Bezug auf Avionik-Software geäußert werden. Betrachtet man andere Arbeiten, die Softwarefehler und deren Folgen untersuchen, stößt man auf die Fallstudie von Leveson (2004). Diese Studie beschäftigt sich mit sechs Unfällen aus dem Bereich der Raumfahrt. Die Verfasserin hält fest, dass bei jeden der Vorfälle fehlerhafte Software zumindest teilweise eine Rolle gespielt hat. So wurde beispielsweise eine Software von amerikanischen und europäischen Entwicklern unabhängig voneinander programmiert und zum Projektabschluss zusammengefasst, ohne dass kulturelle Gegebenheiten berücksichtigt wurden oder eine entsprechende Kontrollen stattfanden. In diesem Fall benutzten die amerikanischen Entwickler zum Beispiel britische Maßeinheiten, wohingegen die Europäer sich auf das metrische System bezogen. Diese Verwechslung hatte zur Folge, dass eine Marssonde ihre Bremstriebwerke zu spät zündete und auf der Planetenoberfläche zerschellte (Leveson, 2004). Durch diesen

Softwarefehler entstanden ein Schaden in Höhe von rund 125 Millionen Dollar und Verzögerungen in Forschungsprojekten, welche auf die geplanten Daten zugreifen wollten (Sawyer, 1999).

Diese exemplarische Auflistung von Fehlern zeigt auf, dass Softwarefehler in den unterschiedlichsten Domänen und auf vielen Ebenen auftreten können. In bestimmten Situationen mögen sie für die Nutzer nur ein kleines Ärgernis darstellen, in anderen Situationen können sie aber auch fatale Konsequenzen nach sich ziehen (Banks et al., 2018; Dikmen & Burns, 2017; Fleetwood, 2017; Hevelke & Nida-Rümelin, 2015; Johnston & Harris, 2019; Leveson, 2004).

1.2.2 Wirtschaftliche Auswirkungen von Softwarefehlern

Ein weiterer Aspekt, der in Bezug auf Softwarefehler nicht vernachlässigt werden darf, sind deren wirtschaftliche Auswirkungen. Dass der Softwareentwicklungsprozess im Rahmen des Software Engineerings auch an wirtschaftliche Gegebenheiten gebunden ist und sich diese auf allen relevanten Ebenen eines Projekts bemerkbar machen, wird noch im nachfolgenden Kapitel 2 deutlich beschrieben (Dijkstra, 1972; Naur & Randell, 1968; Sommerville, 2012). Es muss jedoch bereits vorab erläutert werden, welche immensen Wirtschaftsschäden zum Teil durch fehlerbehaftete Programme ausgelöst werden können und weshalb aus diesem Gesichtspunkt die Qualitätssicherung von fundamentaler Bedeutung ist.

Ein klassisches Beispiel für einen massiven wirtschaftlichen Schaden durch Softwarefehler stellt der Absturz der unbemannten Trägerrakete *Ariane 5* bei ihrem Jungfernflug am 4. Juni 1996 dar (Grotelüschen, 2008; Leveson, 2004; Wunderlich-Pfeiffer, 2015). Als Absturzursache wurden Design- und Spezifikationsfehler ausgemacht. Diese resultierten daraus, dass man aus Kostengründen die Software der *Ariane 4* für das neue Projekt ohne entsprechende Anpassungen wiederverwendet wurde (Leveson, 2004). Somit enthielt die Software viele zusätzliche Funktionen, die zwar als nützlich erachtet werden konnten, jedoch bereits für den Betrieb der *Ariane 4* nicht notwendig waren bzw. nicht genutzt wurden. Die mangelhafte Anpassung an die neue Rakete resultierte letztendlich in einem sogenannten *Stack Overflow*, bei dem es sich in der Softwareentwicklung um einen sehr häufigen Fehler handelt. Dabei wird eine zu große Datenmenge in einen zu kleinen Speicher geschrieben, was zur Folge hat, dass das Programm verfälschte Daten erhält oder abstürzt, genauso wie die genannte *Ariane 5*, nur 30 Sekunden nach ihrem Start (Dowson, 1997; Gleick, 1996; Grotelüschen, 2008; Leveson, 2004; Wunderlich-Pfeiffer, 2015). Der Absturz der *Ariane 5* machte eine grundlegende Überarbeitung der Software notwendig und stellte für die ESA einen Prestigeverlust gegenüber anderen Raumfahrtbehörden dar. Der finanzielle Schaden der durch den Absturz verursacht wurde, wird auf eine Höhe von rund 500 Millionen Dollar geschätzt (Grotelüschen, 2008). Leveson (2004, S.15) zieht aus den Geschehnissen um die *Ariane 5* folgendes Fazit: *"The cost of the decision to reuse the Ariane 4 software without rewriting it (in terms of both money and program delays) was much greater than the cost of doing a proper analysis of its safety."*

Abbildung 1.2: Explosion der Ariane 501 (European Space Agency, 1996)

Auch wenn die 500 Millionen Dollar des gescheiterten Jungfernflugs der *Ariane 5* bereits sehr hoch erscheinen mögen, so handelt es sich dabei lediglich um ein anfangs gescheitertes Großprojekt, dessen Fehler behoben werden konnten (Grotelüschen, 2008). Anders verhält es sich, wenn man die Auswirkungen von Softwarefehlern auf verschiedene Volkswirtschaften betrachtet (Fiutak, 2006; Shapiro, 1987; Tricentis, 2017). So weist beispielsweise bereits Shapiro 1987 auf eine Statistik des amerikanischen *National Institute of Standards and Technology* hin, welche belegt, dass bereits damals durch Softwarefehler Kosten von über 60 Milliarden Dollar verursacht wurden. Berücksichtigt man die voranschreitende Digitalisierung, so verwundern auch nicht die Zahlen von Fiutak aus dem Jahr 2006. In seinem Artikel weist er daraufhin, dass zu dieser Zeit europaweit jährlich ein Schaden von rund 150 Milliarden Euro durch fehlerbehaftete Software entstanden ist. Ein aktuelles Beispiel findet sich in einer Untersuchung der Firma Tricentis (2017), in deren Rahmen 606 weltweit erschienene englischsprachige Nachrichtenmeldungen analysiert wurden, in denen es um Kosten ging, die durch Softwarefehler verursacht wurden. Dabei zeigte sich, dass insgesamt 3,7 Milliarden Menschen von den Fehlern betroffen waren, 314 Firmen von diesen beeinflusst worden sind und sich der jährliche Schaden weltweit auf 1,715 Billionen Dollar beläuft (Tricentis, 2017).

1.3 Thematische Relevanz

Liest man das zu Beginn genannte Zitat *"Wir können die moderne Welt ohne Software nicht aufrechterhalten"* von Ian Sommerville (2012, S.28) an dieser Stelle erneut und berücksichtigt die zuvor genannte Relevanz von Software für unser Alltagsleben und unsere Gesellschaft, sowie die genannten Unfälle und wirtschaftlichen Auswirkungen von fehlerbehafteten Programmen, so mutet seine Aussage nun weniger überspitzt an. Wenn selbst in Bereichen wie der Raumfahrt, die weithin immer noch als die Spitze der menschlichen Technologie und Forschung gilt (Leveson, 2004; Nivala, Hauser, Mottok & Gruber, 2016), derartige Fehler wie bei der erwähnten Mission der Marssonde oder dem Jungfernflug der *Ariane 5* passieren können, wie lässt sich jemals sicherstellen, dass die Software beim Endverbraucher reibungslos und vor allem sicher funktioniert? Diese Frage gewinnt noch zusätzlich an Bedeutung, da die verantwortlichen Programmierer in Domänen wie dem Software Engineering zunehmend mit immer komplexer werdenden Aufgaben konfrontiert werden und die aktuellen Entwicklungen eher darauf hindeuten, dass sich dieser Trend noch zu intensivieren scheint (Broy, 2006; Busjahn et al., 2015; Soloway & Ehrlich, 1984; Weinberg & Schulman, 1974).

Diese Entwicklungen bleiben natürlich auch den Softwareherstellern nicht verborgen. Folglich bemühen sich diese um geeignete Gegenmaßnahmen, die sicherstellen sollen, dass die Qualität der Software auf einem angemessenen Niveau liegt. Um die dabei gestellten Anforderungen erfüllen zu können, werden seitens des Projektmanagments bei größeren Projekten häufig bis zu 80% des Gesamtbudgets für die Qualitätssicherung aufgewandt. Dabei wird auf eine möglichst frühe Erkennung und Behebung von Fehlern gesetzt. Je früher diese korrigiert werden können, desto günstiger gestaltet sich der gesamte Entwicklungsprozess, da folglich auf langfristige und kostspielige Überarbeitungen verzichtet werden kann (Hoffmann, 2013; Hryszko & Madeyski, 2017; Kneuper, 2014; Maxim & Kessentini, 2016; Rashid & Nisar, 2016; Rezagholi, 2004).

Im Rahmen der Qualitätssicherung gibt es eine Vielzahl von anwendbaren Verfahren, die je nach der zu lösenden Aufgabe und Phase des Entwicklungsprozesses stark variieren. Sowohl aus Sicht der Computerwissenschaften, als auch aus einer psychologischen Perspektive betrachtet, stellen sich dabei vor allem die sogenannten Code Reviews als äußerst komplex dar. Was genau unter Code Reviews zu verstehen ist und welche verschiedenen Vorgehensweisen es bei diesen gibt, wird im nächsten Kapitel ausführlich beschrieben. Es sei jedoch vorweg gegriffen, dass es sich bei diesen um eines der am häufigsten eingesetzten Verfahren zur Qualitätssicherung handelt, in dessen Rahmen eine manuelle (jedoch meist toolunterstützte) Durchsicht des Quellcodes durch einen oder mehrere Programmierer erfolgt. Das oberste Ziel des Reviews ist dabei die Verbesserung des zu begutachtenden Codes, indem Fehler identifiziert, dokumentiert und ausgebessert werden (Bacchelli & Bird, 2013; Czerwonka, Greiler & Tilford, 2015; Hoffmann, 2013; Kononenko et al., 2016). Diesbezüglich muss bedacht werden, es sich bei Quellcode um ein Artefakt handelt, welches

von Menschen erstellt, gelesen und gepflegt wird. Insofern muss darauf geachtet werden, dass guter Quellcode nicht nur funktional und fehlerfrei sein sollte, er sollte auch verständlich formuliert sein (Buse & Weimer, 2010). Nur so kann beispielsweise sichergestellt werden, dass auch eine langfristige Betreuung der Software durch andere Programmiererinnen und Programmierer sichergestellt wird und zusätzliche Module zu einem späteren Zeitpunkt implementiert werden können. Diese Art der Qualitätseinschätzung kann zwar maschinell unterstütz, aber nicht vollständig automatisiert werden. Die letzte Entscheidung über die zugrundeliegende Qualität trifft in dieser Domäne immer ein Mensch (Schnappinger, Osman, Pretschner, Pizka & Fietzke, 2018).

Aufgrund des wirtschaftlichen Nutzens und ihrer hohen Komplexität, war die Arbeit mit Quellcode bzw. die Suche nach diesem bereits Gegenstand mehrerer wissenschaftlicher Untersuchungen (Busjahn, Bednarik & Schulte, 2014; Busjahn et al., 2015; Turner, Falcone, Sharif & Lazar, 2014). Aus den selben Gründen führten beispielsweise auch Edmundson et al. (2013) eine Studie durch die sich mit der Effizienz eines modernen, toolunterstützten Code-Reviews beschäftigt, bei dem die Probanden Fehler und Schwachstellen im Sourcecode beheben sollen. Es zeigt sich, dass die Versuchspersonen auf einem hohen Niveau arbeiten, jedoch ist kein einziger Proband dazu in der Lage, alle Schwachstellen zu erkennen und zu beheben. In Bezug auf die oben genannten Beispiele für Softwarefehler regen die Ergebnisse von Edmundson et al. (2013) zum Nachdenken am. Scheinbar können Codes trotz zum Teil massiver Toolunterstützung nicht komplett fehlerfrei geschrieben werden. Ähnliche Ergebnisse zeigen sich auch in anderen Studien wieder. Unter Verwendung von moderner Eye-Tracking-Technologie deuten sich darüber hinaus in diesen zum Teil auch erfahrungsbedingte Unterschiede bei der Ausführung von Code Reviews an, welche sich in den Augenbewegungen der jeweiligen Probanden bemerkbar machen (Obaidellah, Al Haek & Cheng, 2018; Sharafi, Soh & Guéhéneuc, 2015).

Das Auftreten von erfahrungsbedingten Unterschieden zwischen Experten und Novizen, sowie die Auswirkung von Erfahrung auf die in den Prozess des Code Reviews involvierten Augenbewegungen machen dieses Verfahren auch aus Sicht der Expertiseforschung interessant. Der aktuelle Forschungsstand zu diesem Thema, welcher im dritten Kapitel ausführlich beleuchtet wird, weist in dieser Hinsicht Lücken auf, die im Rahmen dieser Dissertation näher beleuchtet werden sollen.

Kapitel 2

Theoretischer Hintergrund

Das nachfolgende Kapitel beschäftigt sich mit den theoretischen Grundlagen dieser Arbeit, die einem interdisziplinären Themenfeld entstammen, welches Bereiche aus der Pädagogik, der Psycholgie, der Softwareentwicklung und dem Eye-Tracking abdeckt.

Das erste Unterkapitel siehe 2.1 geht auf die Domäne des Software Engineerings ein. Dabei wird erläutert, was diese Domäne ausmacht und inwiefern sie für die Expertiseforschung von Interesse ist. Schwerpunktmäßig wird dabei auf den modernen Softwareentwicklungsprozess, sowie dessen historische Entwicklung eingegangen, die noch immer spürbare Konsequenzen in Bezug auf die Qualitätssicherung im Software Engineering nach sich zieht. In diesem Kontext wird auch definiert, was in dieser Dissertation unter einem Code Review verstanden wird (siehe Absatz 2.1.2).

Den nächsten größeren Themenpunkt stellt die Expertiseforschung dar (siehe Unterkapitel 2.2). Hier sollen die thematisch relevanten Annahmen und Theorien erläutert werden. Weiterhin wird dabei darauf eingegangen, wie genau sich das Expertenbild im Kontext von Software Engineering zusammensetzt und welche Trainingsprozesse im Rahmen des Expertiseerwerbs relevant sind (Hauser, Stark, Mottok, Gruber & Reuter, 2020) .

Zwar als Teil der Expertiseforschung, jedoch exponiert hervorgehoben wird anschließend das Thema visuelle Expertise in einem eigenen Abschnitt dieses Kapitels (siehe diesbezüglich 2.3) behandelt. Diese stellt den zentralen Begriff dieser Arbeit dar. Insofern widmet sich dieser Abschnitt einer fundierten Herleitung und Übertragung des Begriffs aus anderen Domänen auf den Kontext der hier vorliegenden Code Reviews.

Abschließend wird auf die Physiologie des menschlichen Auges, sowie die Eye-Tracking-Technologie eingegangen, welche in dieser Dissertation als primäre Erhebungsmethode eingesetzt wird (siehe Unterkapitel 2.4.1). In diesem Rahmen soll kurz die grundlegende Methodik näher erklärt und herausgearbeitet werden, welche Metriken im Kontext von Software Engineering Anwendung finden.

2.1 Die Domäne des Software Engineerings

Dieses Unterkapitel beschäftigt sich mit der Domäne des Software Engineerings, und weshalb diese den thematischen Kontext der vorliegenden Arbeit darstellt. Um den Begriff des Software Engineerings richtig einordnen zu können, muss dieser jedoch erst von der Informatik abgegrenzt werden. Beim Software Engineerings handelt es sich um eine technische Disziplin, welche sich mit allen Aspekten der Softwareherstellung beschäftigt. Der Themenbereich erstreckt sich somit von der Erstellung der Systemspezifikationen über den eigentlichen Entwicklungsprozess, bis hin zur Wartung nach der Inbetriebnahme der Software. Die Informatik konzentriert sich hingegen vorwiegend auf die theoretischen Grundlagen und weist insgesamt deutlich weniger Praxisbezug auf als das Software Engineering (Sommerville, 2012).

2.1.1 Eine kurze Geschichte des Software Engineerings

Der Begriff und das Verständnis von Software Engineering oder Softwareentwicklung durchlebten im Wandel der Zeit einige Veränderungen. Um unsere heutige Auffassung dieser Domäne und die in ihr agierenden Personen genau verstehen zu können, ist ein kurzer historischer Rückblick notwendig (Abran, Moore, Bourque & Dupuis, 2014; Dijkstra, 1972; Langer, 2016; Naur & Randell, 1968; Sommerville, 2012).

Die Ursprünge des Software Engineerings

Software Engineering als Domäne blickt inzwischen auf mehr als 50 Jahre Geschichte zurück. Die gedanklichen Ursprünge des Software Engineerings lassen sich bis in die 1950er- und 1960er-Jahre zurückverfolgen. Zur damaligen Zeit galt im IT-Sektor die Hardware als besonders fehleranfällig und stellte im Regelfall bei Arbeitsprozessen das schwächste Glied in der Kette dar. Während der oft mehrstündigen Berechnungen konnte es durchaus passieren, dass der eingesetzte Computer einen Hardwaredefekt erlitt, ausfiel und repariert werden musste (Abran et al., 2014; Dijkstra, 1972; Embedded Software Engineering, 2018; Langer, 2016; Naur & Randell, 1968; Sommerville, 2012). Was aus der heutigen Perspektive etwas verwunderlich oder skurril wirken mag, stellte in der damaligen Zeit ein derart gravierendes Problem dar, dass als Folge die eingesetzten Computer möglichst ausfallsicher konstruiert und die Algorithmen auf eine Art geschrieben werden mussten, die mit diesen Eventualitäten zurecht kam (Ludewig & Lichter, 2013; Sommerville, 2012). In etwa zur selben Zeit machte auch Gordon Moore, ein Gründungsmitglied der Firma Intel, eine Entdeckung, die später als *Moore's Law* bekannt werden sollte (G. E. Moore, 1965). Er stellte fest, dass die Kenngrößen der Hardware nicht nur zunehmen, sondern ein exponentielles Wachstum aufweisen. Es zeigte sich beispielsweise, dass sich die Prozessorleistung oder die Speicherkapazität pro Dollar im Schnitt alle 18 Monate (je nach Quelle wird alternativ auch von 12 oder 24 Monaten gesprochen) verdoppelt (Ludewig & Lichter, 2013; G. E. Moore, 1965). Auf der Grundlage dieser Berechnung zeichnete sich ab, dass leistungsfähigere Hardware in einem überschaubaren

Zeitraum zur Verfügung stehen würde. Parallel dazu wurde Software zunehmend für die breite Masse interessant, was zur Folge hatte, dass die Nachfrage nach Softwareprodukten anstieg. Diese verstärkte Nachfrage hatte jedoch auch eine negative Begleiterscheinung für die aufstrebende Softwarebranche: Es zeigten sich größere Schwachstellen im bis dahin praktizierten Entwicklungsprozess. Wurde die Hardwareentwicklung anfangs als die zentrale Schwachstelle betrachtet, so verlagerte sich die Problematik Mitte der 1960er-Jahren zunehmend auf die Softwareentwicklung (Sommerville, 2012).

Bis zu diesem Zeitpunkt wurden in der Softwareentwicklung vornehmlich individuelle Ansätze verfolgt. Dabei wurden zum Beispiel ganze Programme von einer einzigen Person oder kleineren Teams für einen ganz speziellen Verwendungszweck vollständig. Unter Anwendung der bis dahin gültigen Paradigmen konnten manche Softwareprojekte selbst mit dem größtmöglichen Aufwand nicht zufriedenstellend verwirklicht werden. Im Entwicklungsprozess wurden häufig die Deadlines und Finanzbudgets signifikant überschritten. Hinzu kam auch noch, dass seitens der verantwortlichen Programmierer vieles für machbar gehalten wurde, was sich jedoch in Wahrheit als nicht realisierbar erwies. Daraus resultierte, dass die entwickelte Software häufig fehlerbehaftet und nur eingeschränkt nutzbar war. Umso größer und komplexer die Programme wurden, umso mehr zeigte sich auch, dass die individuelle Vorgehensweise für die Softwareentwicklung ungeeignet war und die von der Industrie gewünschten Projekte auf diese Art nicht umgesetzt werden konnten. Eine Übertragung oder Anpassung der bisher verwendeten Ansätze war ebenfalls nicht möglich. Die gesamte Softwareentwicklung steckte in einer Krise und es musste dringend nach Lösungsansätzen gesucht werden (Dijkstra, 1972; Hoffmann, 2013; Ludewig & Lichter, 2013, 2010a, 2010b; Naur & Randell, 1968; Sommerville, 2012). In die Geschichte ging dieser Zeitabschnitt als *Software Crisis* ein und steht in der Softwareentwicklung noch immer synonym für die zuvor beschriebene Problematik (Dijkstra, 1972; Hoffmann, 2013; Naur & Randell, 1968).

Die *Software Crisis* und wie sie überwunden wurde

Um die *Software Crisis* (Dijkstra, 1972) zu überwinden und die Softwareentwicklung zeitgemäßer zu gestalten, wurde Ende der 1960er-Jahre intensiv nach möglichen Lösungen gesucht. Dazu wurden beispielsweise verschiedene Studiengruppen gegründet und wissenschaftliche Konferenzen abgehalten (Hoffmann, 2013; Sommerville, 2012). Im Rahmen einer NATO-Konferenz im Jahr 1968 taucht erstmals der Begriff *Software Engineering* auf. Dieser wurde von den verantwortlichen Personen unter der folgenden Begründung bewusst etwas provokant gewählt, da sie darauf aufmerksam machen wollten, was genau der Softwareentwicklung als Disziplin fehlt (Naur & Randell, 1968):

"The phrase software engineering was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering." (Naur

& Randell, 1968, S.8)

Abbildung 2.1: Foto der von Naur und Randell (1968) beschriebenen NATO-Konferenz aus dem Artikel *Raus aus der Software-Krise: 50 Jahre Software Engineering* (übernommen aus Embedded Software Engineering (2018)

Naur und Randell (1968) attestierten in ihrem Konferenzband der Softwareentwicklung der 1960er-Jahre das Fehlen von fundierten theoretischen Grundlagen und eine mangelhafte Standardisierung. Einen möglichen Ausweg sahen sie darin, sich an den Vorgehensweisen anderer ingenieurswissenschaftlicher Bereiche zu bedienen und diese in geeigneter Form an die Bedürfnisse der Softwareentwicklung anzupassen (Hoffmann, 2013; Naur & Randell, 1968; Sommerville, 2012). Basierend auf dieser Annahme war der Begriff Software Engineering geboren, der im weiteren Verlauf des Konferenzbands genauer spezifiziert wurde. Im Wesentlichen zeichneten sich dabei bereits noch heute gültige Merkmale des Software Engineerings ab. So beschäftigte man sich beispielsweise damit (Naur & Randell, 1968):

- in welcher Beziehung Software zur Hardware eines Computers steht.
- wie Software designed werden muss.
- wie Software produziert oder implementiert werden soll.
- welche Vertriebsmöglichkeiten es für Software gibt.
- wie sich die Wartungsmöglichkeiten für Software gestalten.

Das moderne Verständnis von Software Engineering als Domäne

Der Großteil der zuvor beschriebenen Themen ist immer noch fester Bestandteil des Software Engineerings und spiegelt sich auch in der entsprechenden Definition in Ian Sommervilles (2012, S.33) Standardwerk wieder: *"Beim Software Engineering handelt es sich um eine technische Disziplin, die alle Aspekte der Softwareherstellung abdeckt. Das Tätigkeitsspektrum erstreckt sich dabei von den frühen Phasen der Systemspezifikationen*

über die eigentliche Entwicklung der Software bis hin zur Wartung des fertigen Systems, nachdem dieses seine Arbeit aufgenommen hat." Sommerville (2012) ist dabei wichtig zu betonen, dass seine Definition zwei zentrale Schlüsselbegriffe aufweist. Dabei handelt es sich um die *technische Disziplin* und *alle Aspekte der Softwareherstellung*:

- *Technische Disziplin:* Unter diesem Schlüsselbegriff versteht Sommerville (2012) vorwiegend ein methodisches, theoriegeleitetes Vorgehen, so wie dieses auch in anderen technischen Domänen beobachtbar ist (beispielsweise in der Architektur oder im Maschinenbau). Auf der einen Seite ermöglicht diese Vorgehensweise ein strukturiertes Arbeiten, offeriert aber gleichzeitig auch die notwendigen Werkzeuge an die jeweiligen Akteure, mit denen sich sowohl bekannte, als auch bisher unbekannte Probleme lösen lassen. Weiterhin ist unter diesem Begriff auch gemeint, dass im Software Engineering mit organisatorischen und finanziellen Einschränkungen gearbeitet werden muss.
- *Alle Aspekte der Softwareentwicklung:* Der Gegenstandsbereich des Software Engineerings liegt nicht nur auf den technischen Prozessen, sondern umfasst auch weitere Aktivitäten, wie etwa die Projektverwaltung, sowie die Entwicklung von Werkzeugen, Methoden und Theorien, welche für den Entwicklungsprozess relevant sind.

Wie sich an Sommervilles (2012) Definition von Software Engineering zeigt, wirken die im Rahmen der *Software Crisis* (Dijkstra, 1972; Naur & Randell, 1968) ausgearbeiteten Vorgehensweisen nachhaltig auf den Prozess der Softwareentwicklung aus (Abran et al., 2014; Langer, 2016; Lee, 2019; Madesh, 2015; Masood, 2020; Sharma, 2016; van der Hoek, 2018). Wie dieser genau abläuft und welche Elemente im Zuge der vorliegenden Dissertation genauer betrachtet werden, wird im nächsten Abschnitt näher behandelt.

2.1.2 Code Reviews als Teil der Qualitätssicherung im Software Engineering

Die bereits im Titel der Dissertation angesprochenen Code Reviews sind bereits seit Dekaden ein essentieller Bestandteil der Qualitätssicherung in der Softwareentwicklung (Abran et al., 2014; Baum, Liskin, Niklas & Schneider, 2016; Baum, Leßmann & Schneider, 2017; Beller, Bacchelli, Zaidman & Juergens, 2014; Shapiro, 1987; Soloway & Ehrlich, 1984; Sommerville, 2012; Tufano, Pascarella, Tufano, Poshyvanyk & Bavota, 2021; Witte, 2016). Wie auch in vielen anderen Bereichen dieser Dissertation (beispielsweise in der Beschreibung der visuellen Expertise bei 2.3), verhält es sich auch mit dem Begriff der Code Reviews so, dass für diesen zahlreiche Definitionen existieren und ein für die nachfolgende Arbeit gültiges Verständnis in Form einer Arbeitsdefinition geschaffen werden soll (Badampudi, Britto & Unterkalmsteiner, 2019; Baker, 1997; Baum et al., 2016, 2017; Beller et al., 2014; Carullo, 2020; Czerwonka et al., 2015; Indriasari, Luxton-Reilly & Denny, 2020; Mäntylä & Lassenius, 2009; Sommerville, 2012; Tufano et al., 2021; Unkelos-Shpigel & Hadar, 2015; Witte, 2016).

Ein grundlegendes Verständnis für Reviews in der Softwareentwicklung im Allgemeinen schafft Sommerville (2012, S.719-724) in seinem bekannten Lehrbuch *Software Engineering*: *"Reviews und Inspektionen sind Aktivitäten der Qualitätssicherung, mit denen sich die Qualität von Projektlieferungen prüfen lässt. Dabei werden unter anderem die Software, ihre Dokumentation sowie die Prozessprotokolle untersucht, um Fehler und Versäumnisse festzustellen und um zu sehen, ob die Qualitätsstandards eingehalten wurden."* (Sommerville, 2012, S.719) Er präzisiert im Anschluss daran noch die Vorgehensweise für die Ausführung der Reviews (Sommerville, 2012, S.719): *"Bei einem Review überprüfen mehrere Personen die Software und die dazugehörige Dokumentation und suchen nach potenziellen Problemen und fehlender Übereinstimmung mit den Standards. Auf der Basis dieser Ergebnisse beurteilt das Review-Team das Qualitätsniveau eines Systems oder einer Projektlieferung. Anschließend können Projektleiter diese Bewertungen für ihre Planungsentscheidungen oder dazu heranziehen, dem Entwicklungsprozess Ressourcen zuzuweisen."*. Auch wenn Sommerville (2012) eine eher allgemeine Erklärung für Code Reviews in der Softwareentwicklung gibt, so deckt sich sein Verständnis mit den Definitionen anderer Autoren (beispielsweise mit Baum et al.(2016), Baum et al. (2017), Beller et al. (2014) oder Tufano et al. (2021)). Besonders zu beachten ist bei Sommervilles (2012) Beschreibung, dass er die Reviews nicht als ein Tool zur Leistungsmessung betrachtet. Für ihn stellen sie ein essentielles Verfahren zur Qualitätssicherung dar.

Weitere Definitionen für Code Reviews finden sich in der Arbeit von (Beller et al., 2014, S.202): *"Code review is a widely agreed-on best practice in Software Engineering [...], consisting in the manual assessment of source code by human reviewers. It is mainly intended to identify defects and quality problems in the source code before its deployment in a live environment."* Die Forscher (Beller et al., 2014) beziehen sich in ihrer Definition auf einen Artikel von Boehm und Basili (2001), welche sich bereits mit der Wirksamkeit von Code Reviews beschäftigt hat. Gleichzeitig betonen sie den Aspekt, dass diese von Menschen durchgeführt werden explizit ihrer Arbeit ''Code review is the manual assessment of source code by humans, mainly intended to identify defects and quality problems" (Beller et al., 2014, S.202).

Ähnlich verhält sich auch das Verständnis von (Tufano et al., 2021, S.163): *"Code Review is the process of analyzing source code written by a teammate to judge whether it is of sufficient quality to be integrated into the main code trunk."* (Tufano et al., 2021) betonen in ihrer Definition auf der einen Seite, dass die Qualität des Codes durch die im Review verankerten Entscheidungsprozesse bewertet wird, welche im Normalfall durch Teamkolleg*innen durchgeführt werden. Die Bedeutung für die Qualitätssicherung unterstreichen sie weiterhin indem sie darauf verweisen, dass Code, welcher ein Review durchlaufen hat eine deutlich niedrigere Wahrscheinlichkeit aufweist, fehlerhaft zu sein.

Fasst man die zuvor genannten Definitionen (Beller et al., 2014; Boehm & Basili, 2001; Sommerville, 2012; Tufano et al., 2021) zusammen und konzentriert sich auf die Gemeinsamkeiten bzw. Übereinstimmungen, so sind sich die Autorinnen und Autoren in mehreren Punkten einig: Bei Code Reviews handelt es sich um manuel-

le Prozesse, welche von einer oder mehreren Personen in verschiedenen Formaten durchgeführt werden. Diese verfolgen vorrangig das Ziel, die Qualität eines Softwareproduktes zu erhöhen, indem Fehler identifiziert und behoben werden. Dieses Verständnis soll auch in in dieser Dissertation als Arbeitsdefinition genutzt werden. Zur Präzisierung sollte diese Definition noch von Sommerville (2012) abgegrenzt werden. Sommerville (2012) führte an mehreren Stellen die Bedeutung eines Teams für den erfolgreichen Ablauf eines Reviews an. Dies kann so in der Realität definitiv bestätigt werden, ist jedoch mit dem Gedanken von individueller Expertise (welche zentraler Bestandteil dieser Dissertation ist und nachfolgend bei 2.2 näher erläutert wird) bzw. Leistungsmessung nicht vollends vereinbar. Aus diesem Grund liegt der Arbeitsdefinition eher das Verständnis einer individuellen Tätigkeit zugrunde.

2.2 Expertise

Die folgenden Absätze beschäftigen sich mit dem wissenschaftlichen Verständnis von Expertise. Dabei soll geklärt werden, was genau unter dem Begriff verstanden wird (siehe Absatz 2.2.1), wie Personen zu Experten in einem bestimmten Bereich werden können (siehe Absatz 2.2.2), was Expertinnen und Experten kennzeichnet 2.2.3 und was die Expertiseforschung charakterisiert beziehungsweise welcher Methodik sie sich bedient (siehe Absatz 2.2.4).

2.2.1 Der Begriff der Expertise

Um den Begriff der Expertise zu definieren, soll an dieser Stelle auf den Ansatz von Ericsson und Towne aus dem Jahr 2010 zurückgegriffen werden. Die beiden Forscher näherten sich einer Beschreibung des Begriffs an, indem Sie verschiedene lexikalische Definitionen auflisteten und mit der wissenschaftlichen Definition verglichen. In ihrem Fall handelte es sich bei den lexikalischen Quellen um die Microsoft Encarta (2009) und das Merriam-Webster Online Dictionary (2020). Für die vorliegende Betrachtung im Rahmen der Dissertation soll noch der Duden (2023) hinzugenommen werden, welcher den deutschsprachigen Raum abdeckt.

Der Duden (2023) sieht Expertise als *"Fachkenntnis"* oder *"spezielles Wissen"*. Auch wenn diese drei Wörter auf den ersten Blick etwas knapp wirken mögen, so verdeutlichen sie doch, dass das Wissen oder Kenntnisse über einen speziellen Bereich eine entscheidende Rolle im Konzept der Expertise spielen (Ericsson & Towne, 2010; Ericsson, 2016; Gruber, 2007). Relativ ähnlich, aber schon unter Einbeziehung der *Expertenfähigkeiten* sieht die Microsoft Encarta Expertise. Als Kurzdefinition wird dort hinsichtlich Experitse der *"Besitz von Expertenwissen und/oder -fähigkeiten"* angegeben. Somit unterstreicht diese Definition auf der einen Seite die Bedeutung von Wissen, merkt aber auch an, dass Experten über bestimmte Fähigkeiten verfügen, die sie von normalen Menschen unterscheiden (Microsoft, 2009). Auch diesen Fähigkeiten kommt hinsichtlich einer Definition von Expertise eine besondere Rolle zu, so konnten bereits in Bereichen wie der Musik und im Sport gemessen werden, dass sich

signifikante Unterschiede zwischen Experten und Novizen ergeben (Ericsson, Krampe & Tesch-Römer, 1993; Gruber & Lehmann, 2008; Williams, Ford, Hodges & Ward, 2018). Die letzte bedeutende Komponente steuert das Merriam-Wbster Online Dictionary (2020) bei. So findet sich zu Expertise dort folgender Eintrag: *"[H]aving, involving, or displaying special skill or knowledge derived from training or experience"* (Merriam-Webster, 2020). Diese Definition betont, dass Expertise kein Zufallsprodukt ist. Sie ist vielmehr das Resultat von Training und domänenspezifischer Erfahrung und wird von Personen erworben (Ericsson et al., 1993; Gruber, 2007; Gruber & Lehmann, 2008).

Um sich von den Lexikoneinträgen zu lösen und den Bezug zur Forschung herzustellen, greifen Ericsson und Towne (2010) auf die Definition von Simon und Chase (1973) zurück, welche sich mit Expertise im Schach beschäftigt. Die beiden Forscher (Simon & Chase, 1973) stellten in ihrer Studie fest, dass sich Experten in einigen Domänen gravierend von Novizen und weniger erfahrenen Personen unterscheiden. Diese Unterschiede gehen auf die Ausbildung von Mechanismen zurück, die durch das Anhäufen von fachspezifischen Erfahrungen entstehen.

Die Vorgehensweise von Ericsson und Towne (2010) gibt nicht nur einen fachlichen Überblick darüber, was sich hinter dem Begriff der Expertise verbirgt, sie zeigt auch recht deutlich auf, dass das wissenschaftliche Verständnis von Expertise in einigen Punkten vom alltäglichen Verständnis abweicht oder in manchen Fällen sehr vage ist (Schneider, 1993). Schneider sprach dieses Problem bereits in seinem 1993 veröffentlichten Artikel *"Acquiring expertise: Determinantes of exceptional performance"* an. Bis zum heutigen Tage hat sich die Domäne der Expertiseforschung zwar deutlich weiterentwickelt, allerdings zeigt beispielsweise die Betrachtung der drei zuvor genannten lexikalischen Quellen (Duden, 2020; Merriam-Webster, 2020; Microsoft, 2009), dass in Bezug auf die Forschung eine engere Sichtweise auf Expertise angewandt werden muss.

Ein grundlegendes Verständnis für Expertise aus einer wissenschaftlichen Perspektive findet sich in der eben erwähnten Veröffentlichung von Schneider (1993). Er fasst den damaligen Kenntnisstand zusammen und gibt beschreibt Expertise folgendermaßen: *"[E]xpertise refers to extreme or exceptional performance"* (Schneider, 1993, S.313). Schneider betont mit seiner Definition vor allem den Aspekt, dass Expertise mit herausragenden Leistungen einhergeht.

Genauer beschreibt Ericsson (2006c) den Begriff im Vorwort zur ersten Auflage seines *The Cambridge Handbook of expertise and Expert Performance*: *"Expertise [...] refers to the characteristics, skills, and knowledge that distinguish experts from novices and less experienced people"* (Ericsson, 2006b, S.3). Sein Verständnis von Expertise bezieht sich vor allem auf die Unterschiede zwischen Experten und Novizen, welche einer die Hauptuntersuchungsgegenstände der Expertiseforschung sind und auch im Kontext dieser Dissertation von großem Interesse sind (Ericsson et al., 1993; Ericsson & Lehmann, 1996; Ericsson, 2006b; Gruber, 2007; Williams et al., 2018). Weiterhin betont Ericsson (2006) auch, dass sich Experten hinsichtlich ihrer Charakteristiken, ihrer Fähigkeiten

und ihres Wissens von Novizen unterscheiden.

Eine umfassende Definition von Expertise im deutschsprachigen Raum findet sich bei Grubers 2007 publizierten Artikel *Bedingungen von Expertise*. Gruber sieht Expertise als eine domänenspezifische Kompetenz, welche prinzipiell erlernbar ist und bei deren Erlangung Wissen eine bedeutende Rolle spielt. Aufgrund ihres domänenspezifischen Charakters lässt sie sich nicht (oder nur zum Teil) auf andere Domänen übertragen (Ericsson & Lehmann, 1996; Ericsson & Towne, 2010; Ericsson, 2016; Gruber, 2007; Hacker, 1992; Mandl & Spada, 1988).

Basierend auf den zuvor genannten Definitionen wird Expertise im Rahmen dieser Definition als eine domänenspezifische Kompetenz gesehen, welche im Rahmen eines langjährigen Prozesses (siehe diesbezüglich den nachfolgenden Abschnitt 2.2.2 zum Thema Expertiseerwerb) von Personen erlangt wird. Die im Rahmen des Expertiseerwerbs aufgebauten Fähigkeiten, sowie das angesammelte Wissen ermöglichen es den Experten in ihrer Domäne Höchstleistungen zu vollbringen (Ericsson, 2006b, 2016; Ericsson et al., 1993; Ericsson & Towne, 2010; Gruber, 2007; Schneider, 1993).

2.2.2 Expertiseerwerb

Bereits die lexikalischen Definitonen bei 2.2.1 (Duden, 2020; Merriam-Webster, 2020; Microsoft, 2009) haben verdeutlicht, dass Expertise kein Zufallsprodukt ist und von einer Person erworben werden muss. Wie dieser Expertiseerwerb aus wissenschaftlicher Sicht aussieht und was es diesbezüglich für den Kontext der vorliegenden Dissertation zu beachten gilt, soll nachfolgend erläutert werden.

Einen ersten Anhaltspunkt für die zeitliche Dauer des Expertiseerwerbs findet man bei Simon und Chase (1973). Die beiden Forscher konnten belegen, dass es mindestens zehn Jahre dauert, um in einem bestimmten Bereich Expertise zu erlangen. Der Zeitpunkt, wann diese erreicht wird, ist jedoch variabel. So erreichen im Sport die meisten Athleten ihre Höchstleistung im Alter zwischen 20 und 30 Jahren (Müller & Weichert, 2012; Williams et al., 2018). In eher kognitiv geprägten Domänen ist dies um rund ein Jahrzent verschoben. So vollbringen Autoren, Komponisten und Wissenschaftler ihre größten Leistungen meistens im Alter zwischen 30 und 40 Jahren (Ericsson & Towne, 2010; Gruber, 2007; Gruber & Lehmann, 2008; Simon & Chase, 1973). Gemein bleibt allen Domänen, dass im Schnitt mindestens zehn Jahre des exzessiven Übens zur Erlangung von Expertise erforderlich sind (Ericsson et al., 1993; Ericsson & Towne, 2010; Ericsson, 2016; Gruber, 2007; Simon & Chase, 1973). Eine schematische Darstellung des Expertiseerwerbs findet sich in der Abbildung 2.2.

Ericsson et al. (1993) weisen in ihrer bis heute viel beachteten Studie *The role of deliberate practice in the acquisition of expert performance* nach, dass das bloße Anhäufen von domänenspezifischen Erfahrungen oder das bloße Agieren in einer Domäne nicht automatisch eine Leistungssteigerung zur Folge hat (Ericsson et al., 1993; Ericsson & Lehmann, 1996; Ericsson, 2006a, 2006b; Ericsson, Hoffman, Kozbelt & Williams, 2018; Gruber, 2007; Gruber & Lehmann, 2008; Holyoak, 1991). Sie stellten fest, dass dem sogenannten *deliberate practice* eine Schlüsselrolle im Expertiseerwerb zukommt.

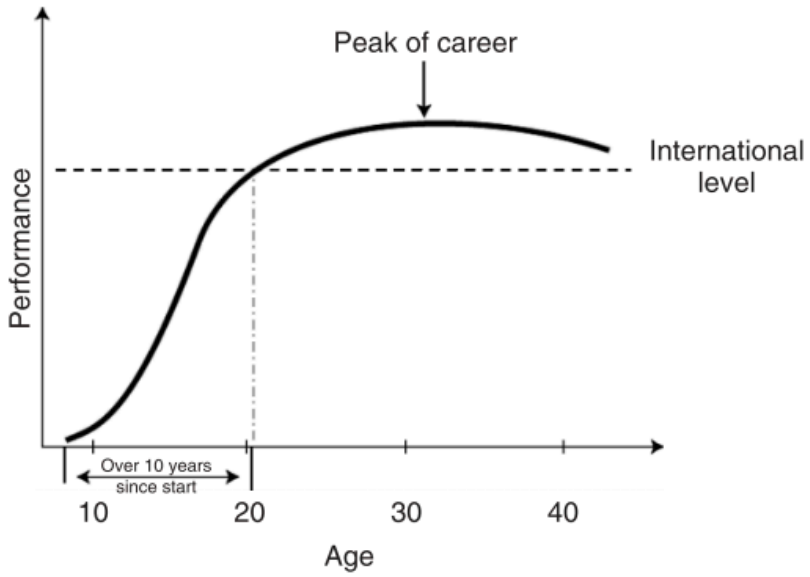

Abbildung 2.2: Zeitlicher Verlauf und Auswirkungen von *deliberate practice* auf die Leistung (übernommen aus der Arbeit von Ericsson und Towne (2010,S.409))

Deliberate practice kann als eine Form des Übens bezeichnet werden, welche zielorientiert auf eine Leistungssteigerung oder die Verbesserung einer bestimmten Fähigkeit ausgelegt ist. Die im *deliberate practice* zu absolvierenden Aufgaben und Übungen sind im Regelfall so gestaltet, dass sie etwas über dem Fähigkeitsniveau der praktizierenden Person liegen und die eigenen Grenzen überschritten werden müssen um ein erfolgreiches Ergebnis zu erzielen. Das Ausführen dieser Aktivitäten wird von den Praktizierenden in den meisten Fällen eher als unangenehm oder zumindest als herausfordernd betrachtet. Idealerweise finden die Trainings unter Anleitung oder Beobachtung eines Trainers statt. Dadurch ist die Möglichkeit gegeben, die jeweilige Aktivität zu wiederholen und gegebenenfalls zu korrigieren oder zu verbessern, falls sich Fehler zeigen (Ericsson et al., 1993; Ericsson & Towne, 2010; Ericsson, 2016; Ericsson et al., 2018; Gruber, 2007; Gruber & Lehmann, 2008). In den meisten Domänen es es notwendig, rund 10.000 Stunden Zeit in *deliberate practice* zu investieren um Expertise zu erlangen. Es zeigt sich weiterhin auch, dass die Performance besser wird, umso mehr Zeit mit *deliberate practice* verbracht wird (Ericsson et al., 1993). Diese Erkenntnis bekräftigt auch, dass Expertise prinzipiell "erlernbar" ist und Training dabei den wichtigsten Faktor darstellt (Ericsson, 2016; Ericsson et al., 2018; Gruber, 2007).

Bezüglich der Übungsprozesse spielen in manchen Domänen die Selbstreflexion und -kritik eine entscheidende Rolle. Diese sind immer dann von Bedeutung, wenn ohne Lehrer oder Trainer geübt wird. Richtig angewandt, ergeben sich in manchen Domänen dabei ebenfalls signifikante Leistungssteigerungen. Zu beachten ist dabei allerdings auch der Übungskontext, bzw. die Art der Übung (Charness, Tuffiash, Krampe, Reingold & Vasyukova, 2005; Ericsson, 2016; Issenberg, McGaghie, Petrusa, Gordon & Scalese, 2005).

Ebenfalls lässt sich beobachten, dass die Erlangung von Expertise stark mit dem Aufbau von Heuristiken und Automatismen verbunden ist. So muss anfangs viel Energie und Konzentration aufgewandt werden um Aufgaben zu lösen. Mit voranschreitendem Expertisegrad wird dieser Aufwand jedoch stark reduziert und Aufgaben können schneller und effizienter erledigt werden (Ericsson, 2016; Ericsson & Towne, 2010; Simon & Chase, 1973).

Viele der Ideen des Expertiseerwerbs und des *deliberate practice* beziehen sich eher auf Domänen wie Sport, Musik, Kunst oder Medizin (Ericsson et al., 1993; Gruber, 2007; Gruber & Lehmann, 2008; Hauser, Stark et al., 2020; Keil, 2016; Müller & Weichert, 2012; Simon & Chase, 1973; van de Wiel, van den Bossche, Janssen & Jossberger, 2011). Parallelen zum Software Engineering bzw. der Softwareentwicklung existieren jedoch ebenfalls: Hauser et al. beschäftigten sich 2020 mit dem Konzept des *deliberate practice* im Rahmen der Softwareentwicklung und konnten eine Reihe von Aktivitäten identifizieren, welche im Sinne von Ericsson et al. (1993) als *deliberate practice* bezeichnet werden können. So zeigt sich beispielsweise, dass Experten Novizen raten würden möglichst viel an sogenannten open-source Projekten teilzunehmen. Durch diese Teilnahme erhalten Novizen die Möglichkeit ihre eigenen Fähigkeiten in einem (relativ) sicheren Umfeld zu testen und weiterzuentwickeln bzw. die Inhalte der formalen Programmierausbildung in der Realität anzuwenden. Gleichzeitig dienen diese Projekte und deren Umfeld für als eine *community of practice*, die soziale Interaktionen und das Einholen von Feedback durch Experten ermöglicht. Gleichzeitig raten Experten auch dazu, die eigenen Fähigkeiten zu verbessern, indem man sich darauf konzentriert eigene Schwächen zu beheben und neue Inhalte beispielsweise in kleinen, gut überprüfbaren Schritten zu erlernen (Hauser, Stark et al., 2020).

Basierend auf dem Vorangegangenen, wird davon ausgegangen, dass die grundlegenden Prinzipien des Expertiseerwerbs auch auf die Domäne des Software Engineerings und auf die Verfahren der Qualitätssicherung (beispielsweise auf die Code Reviews) angewandt werden können (Ericsson et al., 1993; Gruber, 2007; Hauser, Stark et al., 2020). Hinsichtlich dieser Domäne gilt es ebenfalls zu beachten, dass die große Mehrheit der Programmiererinnen und Programmierer eine formale Ausbildung (meist in Form eines Studiums) durchläuft und neben dieser, sowohl durch informelle, als auch durch berufliche Tätigkeiten zusätzliche Erfahrungen sammelt und die eigenen Fähigkeiten ausbaut (Hauser, Stark et al., 2020). Obwohl die Rolle von beruflicher Erfahrung auf den Expertiseerwerb im Software Engineering noch nicht tiefergehend erforscht wurde, scheint diese in einer schnelllebigen Domäne, welche lebenslanges Lernen und die Adaption an neue Gegebenheiten und Techniken voraussetzt, eine nicht zu unterschätzende Rolle zu spielen (Billett, Harteis & Gruber, 2018; Hauser, Stark et al., 2020).

2.2.3 Expertinnen und Experten

Personen, die die im vorherigen Absatz 2.2.3 beschriebenen Übungs- und Lernprozesse erfolgreich absolviert haben, haben (in den meisten Fällen) eine domänen-

spezifische Expertise aufgebaut und werden als Experten betrachtet. Was aus wissenschaftlicher Sicht Experten ausmacht und wie sich dieses Verständnis vom alltäglichen Sprachgebrauch abgrenzt, in dem der Begriff eher inflationär genutzt wird, wird im Anschluss dargelegt.

Auch an dieser Stellle kann auf die Arbeit von Ericsson und Towne (2010) zurückgegriffen werden, in welcher die Autoren versuchen den Begriff des Experten greifbar zu machen. Sie weisen daraufhin, dass populäre Vorstellungen von Expertinnen und Experten typischerweise mit angesehenen Berufen einhergehen, die in vielen Fällen auch mit einem hohen sozialen Prestige verbunden sind. So denkt man bei Experten typischerweise an hervorragende Chirurgen, Luft- und Raumfahrtingenieure oder Richter. Häufig stellt man sich auch noch vor, dass diese Personen bereits seit mehreren Jahrzehnten in ihren Berufen tätig sind (Ericsson & Towne, 2010). All diese Berufe haben gemeinsam, dass sie tatsächlich ein immenses Fachwissen erfordern und in der Regel den erfolgreichen Abschluss eines entsprechenden Studiums voraussetzen. Häufig ist auch noch eine zusätzliche Ausbildung unter Supervision erforderlich, bevor die Personen eigenverantwortlich agieren dürfen (Keil, 2016; van de Wiel et al., 2011). Im Rahmen dieser Ausbildung werden fachspezifische Inhalte ausführlich erklärt, Raum für Fehler wird eingeräumt und relevante Handlungsabläufe werden verinnerlicht, bis diese intuitiv angewandt werden können (Ericsson & Towne, 2010). Damit leiten Ericsson und Towne (2010) bereits über zu den wissenschaftlichen Vorstellungen des Expertenbegriffs. Das wissenschaftliche Verständnis des Begriffs in dieser Arbeit basiert vorrangig auf den Arbeiten der Forschungsgruppen um Ericsson (Ericsson et al., 1993; Ericsson & Kintsch, 1995; Ericsson, 2006b; Ericsson & Towne, 2010; Ericsson, 2016) und Gruber (Gruber, Harteis & Rehrl, 2004; Gruber, 2007; Gruber & Lehmann, 2008) und wird durch entsprechende Erkenntnisse von weiteren Forscherinnen und Forschern (Billett et al., 2018; Charness et al., 2005; Posner, 1988; Williams et al., 2018) ergänzet.

Als Leitlinie zur Erstellung der Arbeitsdefinition wird an dieser Stelle ebenfalls auf die 2007 von Gruber publizierte Arbeit *Bedingungen von Expertise* zurückgegriffen. Gruber (2007) betrachtet Expertinnen und Experten als Personen, welche in einer bestimmten Domäne dauerhaft (also nicht zufällig oder nur ein einziges Mal) herausragende Leistungen vollbringen. Diesen Aspekt der kontinuierlichen Höchstleistung nennt bereits Posner (1988) als ein essentielles Merkmal für die Beschreibung von Expertinnen und Experten. Gleichzeitig spiegelt sich dieser Gedanke auch in den verschiedenen Arbeiten von Ericsson (Ericsson et al., 1993; Ericsson, 2016; Ericsson & Towne, 2010), sowie in vielen Studien aus dem Sport, der Musik und der Medizin wieder (Ericsson & Lehmann, 1996; Keil, 2016; Simon & Chase, 1973; van de Wiel et al., 2011; Williams et al., 2018).

Bezüglich der Definition des Begriffs "Experte" kommt der Rolle von domänenspezifischen Wissen eine wichtige Rolle zu (Ericsson, 2006b; Gruber, 2007). So stützt sich ein hoher Expertisegrads auf eine umfangreiche Wissensbasis, sowie auf reichhaltige Erfahrungen mit domänspezifischen Aufgabenstellungen (Gruber, 2007). Die-

ses angesammelte Wissen bringt eine Reihe von positiven Effekt mit sich. In vielen Domänen verfügen Experten beispielsweise über ein außergewöhnlich gutes Gedächtnis. Dieses weist einen hohen Grad an Strukturiertheit und Organisation auf, wodurch die entsprechenden Personen domänenspezifisches Wissen rasch und weitgehend fehlerfrei aufrufen können (Gruber, 2007; Simon & Chase, 1973). Weiterhin erlauben diese Gedächtnisstrukturen den Expertinnen und Experten auch, neues Wissen und Informationen rasch wahrzunehmen und unmittelbar nach dem Erwerb zu evaluieren und zum erfolgreichen Handeln einzusetzen (Gruber, 2007; Holding, 1985; Simon & Chase, 1973).

Weiterhin zeichnen sich Experten durch die Fähigkeit zu kompetentem Handeln in komplexen (meist beruflichen) Situationen aus (Gruber, 2007; Hacker, 1992). Sie erbringen in ihrer Domäne dauerhaft (also nicht zufällig und nicht nur ein einziges Mal) herausragende Leistungen vollbringen (Gruber, 2007; Posner, 1988). Ein weiteres Markenzeichen stellt diesbezüglich Problemlösefähigkeit dar. Experten gelten in ihren Domänen als sehr gute Problemlöser (Holding, 1985; Gruber, 2007; Krems, 1996). Dies hat mehrere Gründe:

- *Verknüpfung zwischen kognitiven Strukturen und Prozessen:* Die im Expertiseerwerb aufgebauten kognitiven Strukturen (Gedächtnis und Wissen) sind bei Experten eng mit kognitiven Strukturen (Problemlösen und Entscheiden) verbunden (Gruber, 2007). Aufgrund des hohen Strukturiertheitsgrades ihres Wissens können sie dieses schneller und effektiver anwenden (Gruber, 2007; Simon & Chase, 1973).

- *Fallbeispiele und Handlungsroutinen:* Mit zunehmenden Wissen und der gezielten Auseinandersetzung mit einschlägigen Fällen, können sich Experten entsprechende Handlungsroutinen aufbauen (Gruber, 2007). Durch diese Handlungsroutinen können Sie ohne einen größeren Aufwand an Informationsverarbeitung eine Vielzahl von Fälle lösen (Berliner, 2001). Gleichzeitig eignet sich der Experte aber auch Wissen über verschiedene Ausnahmefälle an. Bei diesen Fällen handelt es sich um Konstellationen, in deren Rahmen die Routinen nicht angewendet werden können oder sollen (Gruber, 2007).

- *Ausarbeitung von Lösungsansätzen:* Im Vergleich zu Novizen zeigt sich, dass Experten deutlich mehr Zeit und Anstrengung darauf verwenden, eine anfängliche Problemrepräsentation zu entwickeln. Dabei greifen sie häufig auch auf problemunspezifische Lösungsansätze zurück (Probleme werden beispielsweise umdefiniert und in kleinere Schritte zerlegt). In der Lösungsphase können Experten weiterhin ihren Ansatz bzw. ihr Vorgehen deutlich besser rechtfertigen, da sie diesen bereits vor der Anwendung selbst reflektiert haben und verschiedene Szenarien durchgegangen sind (Holding, 1985; Gruber, 2007; Krems, 1996).

- *Metafähigkeiten:* Verglichen mit Novizen, zeigt sich, dass Experten über bessere Metafähigkeiten verfügen. Sie sind in der Lage ihre eigenen Leistungen zu

planen, zu überwachen und zu evaluieren (Ericsson, 2016).

Letztendlich führt die Kombination aus diesen Merkmalen dazu, dass Experten über bessere Problemlösefähigkeiten verfügen und einen flexibleren Umgang mit domänenspezifischen Problemlöseprozessen demonstrieren (Berliner, 2001; Ericsson, 2016; Holding, 1985; Gruber, 2007; Krems, 1996).

Fasst man die in diesem Absatz dargestellten Definitionen und Merkmale von Experten zusammen, so ergibt sich für den weiteren Verlauf dieser Dissertation die folgende Arbeitsdefinition: *Bei einem Experten handelt es sich um eine Person, welche in ihrer Domäne dauerhaft herausragende Leistungen vollbringt, in dieser über einen umfangreichen und gut strukturierten Wissensstand verfügt, sowie ein fundiertes Problemlöseverhalten an den Tag legt* (Ericsson & Towne, 2010; Hacker, 1992; Gruber, 2007; Posner, 1988). Aus Sicht des Software Engineerings und der bereits im Titel dieser Dissertation erwähnten Code Reviews kommt dem letztgenannten Aspekt des Problemlösens eine starke Bedeutung zu. Wie in Absatz 2.1.2 beschrieben, fokussieren sich die Reviews als Teil der Qualitätssicherung vorrangig darauf eine möglichst hohe Güte des finalen Produkts zu gewährleisten und enthaltene Fehler zu identifizieren und zu lösen. Insofern lässt sich festhalten, dass domänenspezifisches Wissen, sowie fallspezifische Erfahrungen von zentraler Bedeutung sind für das Verständnis des Expertenbegriffs im Software Engineering sind (Berliner, 2001; Ericsson & Towne, 2010; Hoffmann, 2013; Sommerville, 2012).

2.2.4 Expertiseforschung

Aus dem Vorangegangenen zur Expertise hat sich bereits gezeigt, dass der Hauptgegenstandsbereich der Expertiseforschung die Erforschung von Unterschieden zwischen Experten und Novizen ist. Aufgrund des breiten Spektrums, sowie der interdisziplinären Natur der Expertiseforschung hat sich für diese über die Jahre hinweg eine eigene Methodik entwickelt (Ericsson, 2006a, 2016; Ericsson & Towne, 2010), welche nachfolgend für die Domäne des Software Engineerings erläutert werden soll.

Ein grundlegendes Verständnis für den Begriff der Expertiseforschung schafft die Definition von Gruber und Lehmann (2008,S.3): *"Die Expertisetheorie und -forschung nimmt im Kontext entwicklungspsychologischer Arbeiten einen Sonderfall ein. Ihr Ziel ist die Beschreibung, Erklärung und Förderung des Entstehens herausragender Leistungen und Fähigkeiten in jeweils bestimmten Gegenstandsbereichen (Domänen), z. B. Tennis, Klavierspiel oder Schachspiel."*

Die Expertiseforschung unterscheidet prinzipiell zwischen zwei verschiedene Ansätzen. Historisch betrachtet existierte zuerst der *expertise approach*. Dieser betrachtete die individuellen Entwicklungen, die Personen auf ihrem Weg zur Expertise machten. Um einen Experten zu definieren wurde häufig auf selektive Methoden zurückgegriffen, so sollten beispielsweise Kollegen äußern, wen sie als am fähigsten in ihrer Abteilung erachten (Chi, 2006; Ericsson & Towne, 2010; Feltovich, Prietula & Ericsson, 2006). Beim zweiten Erkläransatz handelt es sich um den sogenannten *expert-performance approach*. Dieser beschäftigt sich mit der Struktur und Entwicklung von

überlegenen Leistungen, die reproduzierbar sind (Ericsson, 2006a, 2006b, 2016; Ericsson et al., 1993; Ericsson & Towne, 2010).

Um die Methodik der Expertiseforschung bzw. des *expert-performance approach* zu verstehen, muss man sich vergegenwärtigen, dass diese Ansätze Expertise und die damit verbundenen Eigenschaften in den jeweiligen Domänen erfassen wollen. Dies bedingt bereits Offenheit für interdisziplinäres Arbeiten. Allerdings geht damit auch ein zentrales Problem dieser Disziplin einher: Die Messbarkeit von Expertise bzw. der Expertenleistung (Ericsson, 2016; Ericsson et al., 2018). In bestimmten Domänen, beispielsweise dem Schach oder im Sport (Charness et al., 2005; Gruber & Lehmann, 2008; Simon & Chase, 1973; Williams et al., 2018) mag die Messbarkeit verhältnismäßig einfach sein, da es feste Leistungskriterien gibt, welche erfasst werden können. Anders hingegen stellen sich Domänen wie das Software Engineering (Hauser, Stark et al., 2020), die Medizin (Keil, 2016; van de Wiel et al., 2011) oder die Musik (Ericsson et al., 1993; Gruber & Lehmann, 2008) dar, welche in ihrer Natur freier gestaltet sind und nicht unbedingt von objektiven Kriterien geprägt sind (Ericsson, 2016). Ericsson (2016) bringt an dieser Stelle die Reproduzierbarkeit von Leistungen ins Spiel und sieht diese als einen entscheidenden Faktor wenn es um die Bewertung von Leistungen geht. Dieser Gedanke wird bereits in den oben genannten Erläuterungen zum Verständnis eines Experten (siehe Absat 2.2.3) genannt und von Gruber (2007) weiter vertieft.

Als Methode der Wahl hat sich in der Expertiseforschung über die Jahre hinweg der kontrastive Vergleich zwischen Experten und Novizen etabliert (Chi, 2006; Ericsson et al., 2018, 1993). In diesem werden Experten und Novizen bei der Bearbeitung verschiedener domänenspezifischer Aufgaben gegenüber gestellt und hinsichtlich ihres Vorgehens miteinander verglichen. Aufgrund des Erfahrungsunterschieds ergeben sich zwischen den unterschiedlichen Gruppen Unterschiede, welche mit dem Vorhandensein von Expertise assoziiert werden können (Chi, 2006; Ericsson et al., 2018, 1993).

Als Konsequenz für das Software Engineering und das Studiendesign dieser Dissertation ergibt sich, dass eine Messung von Expertise bzw. der Expertenleistung reproduzierbar sein muss. Ebenso müssen Aufgaben (beispielsweise Code Reviews) so gestellt werden, dass sie sowohl von Experten, als auch von Novizen bearbeitet werden können.

2.3 Visuelle Expertise

Im Vorangegangenen fiel der Begriff visuelle Expertise bereits des Öfteren, wurde dabei aber lediglich am Rand mit Software Engineering oder Code Reviews in Verbindung gebracht. Diese Verbindung soll im nachfolgenden Unterkapitel hergestellt werden.

In Bezug auf das Software Engineering ergibt sich jedoch die Problematik, dass dieses als Domäne noch keine empirisch fundierte Methodik zur Analyse und In-

terpretation von Augenbewegungen während bestimmter Tätigkeiten entwickelt hat und folglich auch keine gesicherte Definition für den Begriff der visuellen Expertise vorliegt (Nivala et al., 2016; Obaidellah et al., 2018; Sharafi et al., 2015). Diese Thematik wird in Kapitel 3, welches sich speziell mit dem Forschungsstand zur visuellen Expertise bei Code Reviews beschäftigt näher erörtert. Um diese Problematik zu lösen wird angeraten (Sharafi et al., 2015), sowohl den Begriff der visuellen Expertise, als auch die dahinterstehenden Theorien aus anderen Domänen abzuleiten und auf den Kontext des Software Engineerings zu übertragen (Obaidellah et al., 2018; Sharafi et al., 2015).

Die nachfolgenden Absätze widmen sich daher dem Versuch, eine passende Definition für visuelle Expertise im Kontext des Software Engineerings zu erstellen, sowie mögliche theoretische Grundlagen genauer zu spezifizieren.

2.3.1 Der Begriff der visuellen Expertise

Sucht man im Software Engineering (oder spezieller im Fall von Code Reviews) nach einer Definition für visuelle Expertise, stellt man fest, dass dieser Begriff zwar häufig benutzt wird, aber im Kontext der Domäne keine entsprechende theoretische Fundierung vorhanden ist (Obaidellah et al., 2018; Sharafi et al., 2015). Sharafi et al. (2015) weisen darauf hin, dass im Software Engineering relativ leichtfertig mit Begriffen aus anderen Domänen umgegangen wird, diese oft aus ihrem Kontext gerissen und ohne entsprechender theoretischer Untermauerung benutzt werden. Sie legen nahe, dass sowohl feststehende Begriffe, sowie etablierte Theorien aus anderen Domänen hinsichtlich ihrer Eignung überprüft und übernommen werden sollten. Dabei verweisen sie vor allem auf die Psychologie und die Medizin (insbesondere die Radiologie), in welchen bereits seit längerer Zeit Studien zur Erforschung visueller Expertise stattfinden (Fox & Faulkner-Jones, 2017; Gegenfurtner et al., 2017; Kok, 2016; Sheridan & Reingold, 2017).

Die ebenfalls von Obaidella et al. (2018) und Sharafi et al. (2015) angesprochene Idee, auf die theoretische Grundlagen und Definitionen anderer Domänen zurückzugreifen und diese auf das Software Engineering zu übertragen ist in ihrer Grundlage nicht neu. Ähnliches berichten auch Sheridan und Reingold (2017) in ihrer Metastudie, welche sich mit visueller Expertise in der medizinischen Bildbetrachtung beschäftigt. Auch sie verweisen explizit darauf, dass derartige Transfers bereits früher erfolgreich durchgeführt wurden und bei einer entsprechenden Passung die untersuchte Domäne von diesen profitieren kann.

Basierend auf dieser Vorgehensweise finden sich erste Anhaltspunkte einer empirisch abgesicherten Definition für visuelle Expertise in der Arbeit von Reingold und Sheridan (2012). In einer Studie, die sich vorrangig mit visueller Expertise beim Schach und in der Medizin beschäftigt, definieren Reingold und Sheridan (2012) den Begriff anwendungsorientiert und verweisen auf die Fähigkeiten, die ihrer Meinung nach typischerweise mit visueller Expertise assoziiert bzw. im Expertiseerwerb aufgebaut werden: *"[...] [O]ne of the most fascinating and impressive aspects of skilled per-*

formance is the ability of the experienced eye to encode at a glance the essence of briefly presented stimulus material that is related to the domain of expertise" (Reingold & Sheridan, 2012, S.524). Das Autorenduo geht mit dieser Definition auf eine der meist genannten Fähigkeiten von Experten in einer visuell geprägten Domäne ein: Im Vergleich zu einem Novizen sind Experten scheinbar dazu in der Lage, einen komplexen und unübersichtlichen Stimuli in Sekundenbruchteilen mit nur einem einzigen Blick zu betrachten und die darin enthaltenen Informationen aufzunehmen (Evans, Haygood, Cooper, Culpan & Wolfe, 2016; Holmqvist et al., 2011; "Diagnostic performance on briefly presented digital pathology images", 2015; Kundel & La Follette, 1972). Gleichzeitig betonen Reingold und Sheridan (2012) auch, dass es sich um eine domänenspezifische Kompetenz handelt, die nicht oder bestenfalls nur teilweise auf andere Bereiche übertragen werden kann.

Sehr ähnlich, jedoch mehr auf die involvierten Wahrnehmungsprozesse und die Kognitionen bezogen, gestaltet sich ein weiterer Ansatz der beiden Autoren (Sheridan & Reingold, 2017). Dieser stammt aus einem Literaturvergleich, welcher schwerpunktmäßig in der Radiologie angesiedelt ist: *"Human visual expertise in many fields reflects complex cognitive and perceptual processing that is developed over the course of many hours of practice and training [...]. In the domain of medical image perception, extensive training and experience is required to learn how to interpret medical visualizations, which are visual images that represent human anatomical structures or functions"* (Sheridan & Reingold, 2017, S.1). Neben den kognitiven Aspekten betont die Definition auch, dass visuelle Expertise das Resultat von intensivem Training ist und nicht durch Zufall entsteht. Vielmehr wird sie durch die exzessive Auseinandersetzung mit einem bestimmten Gegenstandsbereich (sei es eine MRT-Aufnahme, ein Röntgenbild oder im Fall der vorliegenden Dissertation ein Quellcode) über Jahre hinweg aufgebaut und durch das Sammeln neuer Erfahrungen stetig verfeinert. Dieser Fähigkeitserwerb orientiert sich nach Sheridan und Reingold (2017) direkt an den aus der Expertiseforschung bekannten Grundlagen und Parametern und weist deutliche Ähnlichkeit zum bereits zuvor beschriebenen *deliberate practice* auf (Ericsson et al., 1993; Ericsson, 2006a, 2006b; Ericsson et al., 2018). Es sei jedoch angemerkt, dass Sheridan und Reingold (2017) nicht näher auf die Trainingsprozesse eingehen. Somit bleibt offen, ob es sich dabei um routinemäßige Aufgaben handelt, die fester Bestandteil der alltäglichen Arbeit sind oder es sich dabei tatsächlich um eine zielgerichtete Auseinandersetzung im Sinne des *deliberate practice* handelt (Billett et al., 2018; Ericsson, 2006a, 2006b; Ericsson et al., 1993). Die Aussagen der beiden Definition deuten jedoch eher auf Letzteres hin (Reingold & Sheridan, 2012; Sheridan & Reingold, 2017).

Zuspruch erhalten die eben beschriebenen Ansätze (Reingold & Sheridan, 2012; Sheridan & Reingold, 2017) auch von Gegenfurtner et al. (2017), welche visuelle Expertise ebenfalls als das Resultat eines Lernprozesses und als eine domänenspezifische Kompetenz betrachten. Sie definieren sie als *"[...] the superior visual skill shown when executing domain-specific visual tasks"* (Gegenfurtner et al., 2017, S.97). In einer weiteren Arbeit greifen Gegenfurtner und Merriënboer (2017) die Idee auf, dass es

sich bei visueller Expertise um die maximale Anpassung an die Erfordernisse einer stark visuell geprägten Aufgabe handelt.

Deutlich konkreter, sowie mehr auf die Fähigkeiten und das Leistungsvermögen der Expertinnen und Experten bezogen, gestaltet sich die Definition von visueller Expertise seitens Holmqvist et al. (2011): *"A host of studies show that expertise gives rise not only to a more task efficient selection of gaze positions, but also to a superior perceptual processing from a larger functional visual field, stretching further out from the fixation point, than it does for novices [...]"* (Holmqvist et al., 2011, S.396).

Basierend auf den zuvor geschilderten Definitionen und Erklärungsansätzen der verschiedenen Forscherinnen und Forscher (Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Holmqvist et al., 2011; Reingold & Sheridan, 2012; Sheridan & Reingold, 2017) wird der Begriff der visuellen Expertise im Rahmen dieser Dissertation als eine domänenspezifische Kompetenz in der stark visuell geprägten Domäne des Software Engineerings betrachtet. Visuelle Expertise ist das Resultat von Trainingsprozessen und der intensiven Beschäftigung mit einem bestimmten Gegenstandsbereich über einen längeren Zeitraum hinweg. Während dieses Trainings passt sich die praktizierende Person an die Anforderungen der jeweiligen Domäne an wobei die relevanten kognitiven Prozesse der visuellen Informationsaufnahme und -verarbeitung bezüglich ihrer Effizienz optimiert werden.

2.3.2 Modelle zur visuellen Expertise

Ähnlich wie mit dem Begriff der visuellen Expertise stellt sich auch dessen theoretische Fundierung im Software Engineering dar. Wie bereits zuvor geschildert wurde, weist dieses als Domäne zwar eine sehr hohe Strukturiertheit auf (Hoffmann, 2013; Sommerville, 2012), die Auswertung von Augenbewegungen stellt jedoch zum aktuellen Zeitpunkt in vielerlei Hinsicht ein noch unerschlossenes Neuland dar. Sharafi et al. (2015) merken in ihrer Metastudie diesbezüglich an, dass es vielen der analysierten Studien an einer entsprechend geeigneten theoretischen Grundlage mangelt oder diese nicht konsistent verwendet wird. Wie auch in Bezug auf den Begriff der visuellen Expertise, schlägt die Autorengruppe vor, sich dabei auf andere Domänen zu stützen und bei einer passenden Eignung deren Erkenntnisse auf das Software Engineering zu übertragen. In Hinblick auf die Untersuchung von visueller Expertise, werden von Sharafi et al. (2015) vor allem die Psychologie und die Medizin, insbesondere die Radiologie genannt. Diese Domänen untersuchen bereits seit längerer Zeit visuelle Expertise, bieten eine breite Basis von theoretischen Ansätzen und greifen auf diese bei der Durchführung von Studien auf diese zurück (Gegenfurtner, Lehtinen & Säljö, 2011; Kok et al., 2016; Kok, 2016; Sheridan & Reingold, 2017).

Als Musterbeispiel für die Vorschläge von Sharafi et al. (2015) kann die Dissertation von Ellen Kok (2016b) betrachtet werden. Diese beschäftigt sich mit visueller Expertise in der Radiologie. Dabei greift sie auf Theorien zurück, welche sowohl von der Medizin, als auch von der Psychologie akzeptiert werden und ihrem Forschungsvorhaben so eine entsprechend breite Basis verleihen. Bei den von Kok verwendeten

Theorien handelt es sich um das *holistic model of image perception*, die *information reduction theory* und die *theory of long-term working memory*. Diese wurden bereits von anderen Autoren vor ihrer Dissertation mehrfach bei interdisziplinär veranlagten Studien in den Themenbereichen der Medizin (insbesondere in der Radiologie) (Sheridan & Reingold, 2017), dem Schachspiel (Reingold & Sheridan, 2012; B. P. Wood, 1999; G. Wood et al., 2013), sowie bei der Interpretation verschiedener Alltagssituationen (Donovan & Litchfield, 2013) erfolgreich angewandt.

Auch in der hier vorliegenden Arbeit soll primär auf den Ansatz von Ellen Kok (2016b) zurückgegriffen werden. Aufgrund der domänenspezifischen Beschaffenheit von Quellcode, wird Koks Modell (2016b) noch um die *cognitive load theory* (Sweller, 1988, 1994; Sweller, van Merrienboer & Paas, 1998) ergänzt. Diese soll im Sinne der Interpretation der gezeigten Expertenleistungen weitere Möglichkeiten eröffnen und Rückschlüsse auf die Informationsverarbeitung der teilnehmenden Personen ermöglichen. Ebenso sollen durch die Einbeziehung der *cognitive load theory* weitere Aspekte des Software Engineerings (beispielsweise Usability, Coding Guidelines, ... (Hart & Staveland, 1988; Hoffmann, 2013; Sommerville, 2012) abgedeckt werden.

Zusammenfassend legen die Ergebnisse von Kok (2016b) und Sheridan und Reingold (2017), sowie das Nichtvorhandensein von geeigneten Ansätzen zur Analyse von Augenbewegungen im Software Engineering (Obaidellah et al., 2018; Sharafi et al., 2015) legen nahe, dass die genannten theoretischen Grundlagen auch im Zuge dieser Arbeit angewandt werden können. Das Verständnis von visueller Expertise ruht somit auf vier Theorien welche in Abbildung 2.3 veranschaulicht werden. Diese werden in den nachfolgenden Absätzen genauer erläutert.

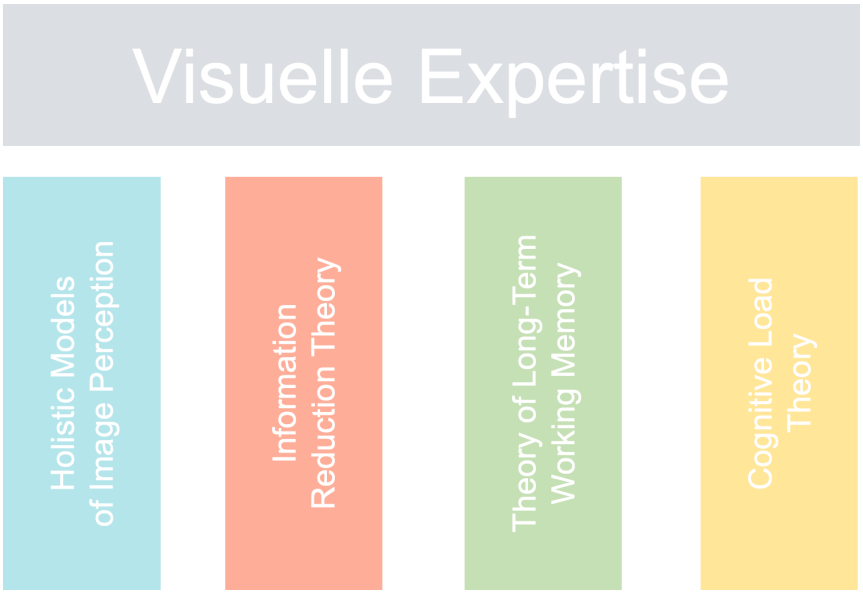

Abbildung 2.3: Erweitertes Modell der visuellen Expertise basierend auf Kok (2016b)

2.3.3 Das *holistic model of image perception*

Das 2016 von Kok benutzte *holistic model of image perception* stellt lediglich ein Modell bzw. eine Zusammenfassung mehrere Modelle zur holistischen Betrachtung von visuellen Stimuli dar. Eine Gegenüberstellung und detaillierte Beschreibung mehrerer Modelle findet sich in einer Metaanalyse von Sheridan und Reingold (2017), welche sich mit visueller Expertise bei der Betrachtung von medizinischem Bildmaterial beschäftigt. Sie führen dabei das *global-focal search model* (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010), das *two-stage detection model* (Swensson, 1980), sowie den Ansatz des *holistic mode vs. search to find* (Kundel, Nodine, Conant & Weinstein, 2007) an und arbeiten dabei deren jeweiliges Alleinstellungsmerkmal, Gemeinsamkeiten und Gegensätze heraus.

Das *global-focal search model*

Das *global-focal search model* wurde ursprünglich 1987 von Nodine und Kundel erstellt und im Laufe der Jahre immer wieder untersucht und modifiziert (Nodine & Mello-Thoms, 2000, 2010). In seiner Grundannahme ist das *global-focal search model* phasenbasiert und kann dazu eingesetzt werden, um die Vorgehensweise von Experten bei der Betrachtung eines visuellen Stimulus zu erklären (Nodine & Kundel, 1987; Sheridan & Reingold, 2017).

In der ersten Phase *global* wird vom Experten ein initialer Scan des zu betrachtenden Stimulus durchgeführt. Der Scan dient primär dazu, sich über den Stimulus einen Überblick zu verschaffen und die darin enthaltenen Informationen aufzunehmen. Während der Betrachtung gleichen Experten die Informationen mit ihrem Vorwissen ab, welche prototypische Vorstellungen von Normalfällen und Abnormitäten enthält. Diese kognitiven Konstrukte werden im Rahmen der Theorie auch als Schemata bezeichnet (Nodine & Kundel, 1987). Durch diesen Abgleich können eventuell vorhandene Auffälligkeiten, Störfaktoren oder Abweichungen bereits identifiziert werden (Nodine & Kundel, 1987; Nodine, Kundel, Lauver & Toto, 1996; Sheridan & Reingold, 2017).

Die nächste Phase *focal* ist dadurch gekennzeichnet, dass die durch den initialen Scan erkannten Auffälligkeiten genauer betrachtet und analysiert werden. Dies macht sich vor allem hinsichtlich der auftretenden Augenbewegungen deutlich. Bei Experten lässt sich beobachten, dass die Fovea (die Region des menschlichen Auges, welche eine genaue Untersuchung und Aufnahme von Informationen ermöglicht) auf die betreffenden Bereiche gelenkt wird, was wiederum signalisiert, dass diesen besondere Aufmerksamkeit zuteil wird (Nodine & Kundel, 1987; Nodine et al., 1996; Sheridan & Reingold, 2017).

Nodine und Mello-Thoms (2000, S. 869) erklären, dass die Prozesse beim *global-focal search model* als eine lineare Abfolge zu verstehen sind, betonen aber gleichzeitig, dass sich diese bei der Betrachtung eines Stimulus wiederholen können: *"attention shifts back to the medical image for a new global impression flagging another perturbed region, focal analysis searches it, a new object may be recognized and recursive testing for abnormali-*

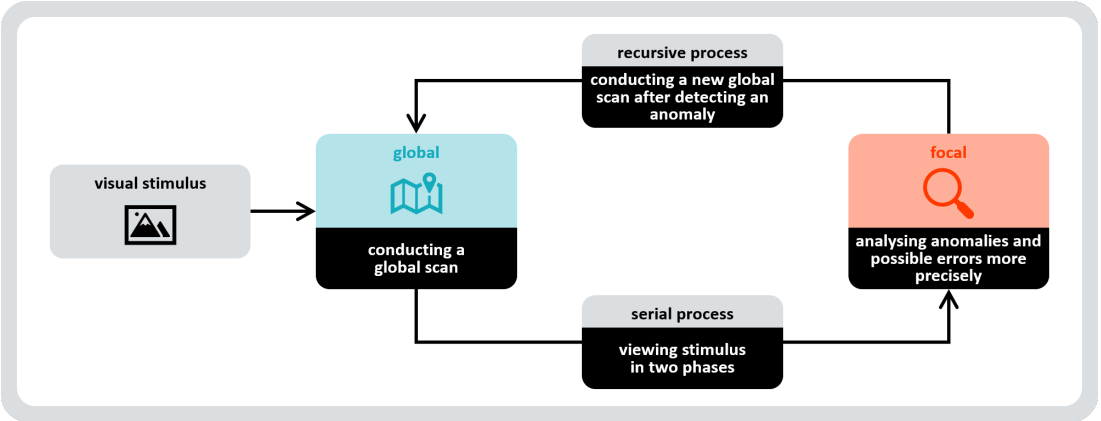

Abbildung 2.4: *Global-focal search model* von Nodine und Kundel (1987), entnommen aus Hauser et al. (2023b)

ties continues until the observer is satisfied that enough evidence has accumulated to make a diagnostic decision." (Nodine & Mello-Thoms, 2000, S.869)

Das Zitat von Nodine und Mello-Thoms (2000, S.869) unterstreicht weiterhin auch die Rolle von Entscheidungsprozessen, die während der Betrachtung eines visuellen Stimulus auftreten können und zeigt gleichzeitig auf, dass beispielsweise Diagnoseprozesse an das Vorwissen der betrachtenden Person gebunden sind. Eine schematische Darstellung des *global-focal search model* findet sich nachfolgend in Abbildung 2.4 wieder.

Das *two-stage detection model*

Das *two-stage detection model* von Swensson (1980) weist einige Ähnlichkeiten zum *global-focal search model* (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010) auf, indem es die Bedeutung einer holistischen Vorgehensweise bei der Bildbetrachtung betont. Beide Modelle stimmen diesbezüglich darin überein, dass Experten dazu in der Lage sind, große Regionen eines visuellen Stimulus schnell zu erfassen, indem sie ihr parafoveales und peripheres Sehen einsetzen um sich einen Überblick zu verschaffen. Auffälligkeiten werden dann durch das foveale Sehen genauer betrachtet (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Swensson, 1980).

Ähnlich zum *global-focal search model* (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010), basiert auch das *two-stage detection model* (Swensson, 1980) auf der Annahme, dass es zwei Verarbeitungsstufen bei der Betrachtung eines visuellen Stimulus gibt. Im Gegensatz zum Erstgenannten, geht dieses Modell jedoch nicht davon aus, dass die involvierten Prozesse rekursiv sind. Swenson (1980) vertritt die Meinung, dass durch die Anwendung der als Filter fungierenden Automatismen in der ersten Phase die Auffälligkeiten identifiziert und ausgewählt werden, welche in der nachfolgenden Phase genauer analysiert werden. Die ausgewählten Auffälligkeiten erhalten in diesem Prozess die volle Aufmerksamkeit von der betrachtenden Person. Im Rahmen der dabei durchgeführten Analyse werden die Auffälligkeiten mit dem Vorwissen abgeglichen. Basierend darauf, werden kognitive Bewertungs-

prozesse durchgeführt und es wird bestimmt ob und mit welchem Vertrauensniveau die Auffälligkeit die zu klärenden Kriterien erfüllt. Eine schematische Darstellung des *two-stage detection models* findet sich in Abbildung 2.5 wieder.

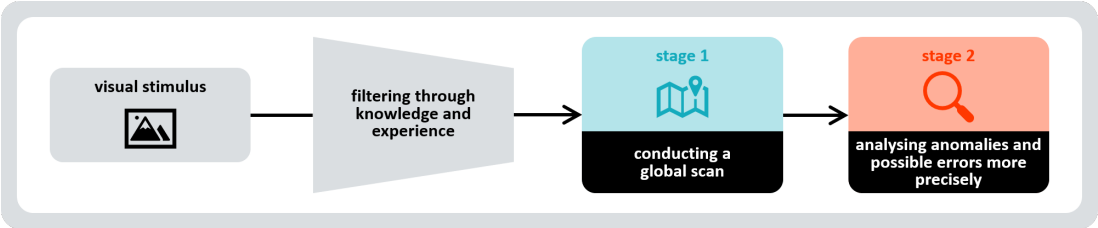

Abbildung 2.5: *Two-stage detection model* von Swensson (1980), entnommen aus Hauser et al. (2023b)

Der Ansatz des *holistic mode vs. search to find*

Einen weiteren Ansatz zur holistischen Betrachtung visueller Stimuli stellt die von Kundel et al. (2007) beschriebene Vorgehensweise des *holistic mode vs. search-to-find* dar (siehe Abbildung 2.6). Kundel et al. (2007) gehen davon aus, dass sich das Erreichen von Expertise in der medizinischen Bildbetrachtung auch dadurch ausdrückt, dass von einem relativ langsamen *search-to-find*-Ansatz zu einer deutlich schnelleren holistischen Vorgehensweise verändert.

Teil der holistischen Vorgehensweise ist es, einen schnellen und umfassenden Scan des zu betrachtenden Bildes durchzuführen. Dieser erlaubt es den Experten, Auffälligkeiten oder Störfaktoren zu identifizieren. Sobald diese entdeckt wurden, wechseln die Experten zur *search-to-find*-Methode und betrachten die zuvor selektierten Regionen und scannen auch den Rest des Bildes um weitere potenzielle Auffälligkeiten zu erkennen, die während des ersten Durchgangs nicht salient genug waren, um identifiziert zu werden. Bezüglich der beiden Vorgehensweisen in diesem Ansatz betonen Kundel et al. (2007), dass diese parallel zueinander ausgeführt werden können. Es können selbst dann noch weitere globale Informationen aufgenommen werden, wenn die betrachtende Person bereits im *search-to-find*-Modus ist und sich mit der Analyse von Auffälligkeiten beschäftigt.

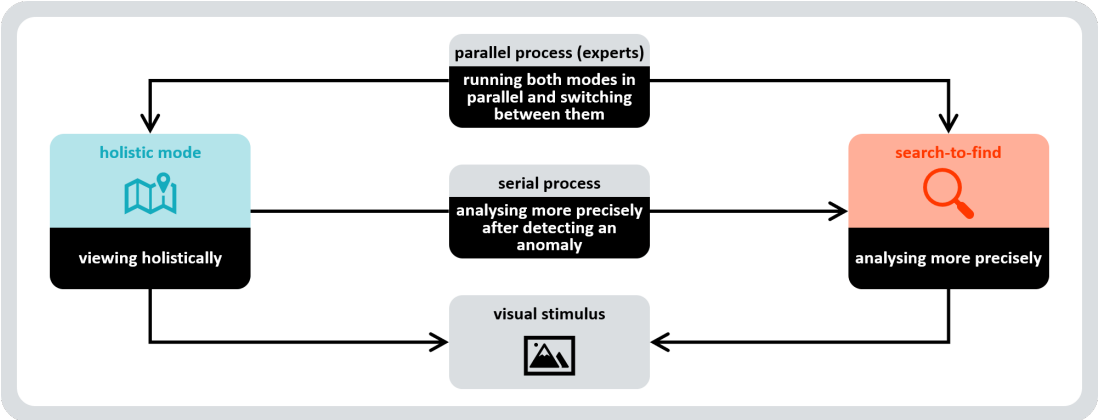

Abbildung 2.6: *Holistic mode vs. search to find* von Kundel et al. (2007), entnommen aus Hauser et al. (2023b)

Kundel et al. (2007) stellten darüber hinaus fest, dass die Fähigkeit zur schnellen holistischen Betrachtung eines visuellen Stimulus an das Training und die Erfahrungen der durchführenden Person gebunden ist. und erst im Lauf des Expertiseerwerbs aufgebaut wird. Aus diesem Grund findet sich bei Novizen im Gegensatz zu Experten vorwiegend der langsamere *search-to-find*-Ansatz, da die Fähigkeit zur holistischen Betrachtung noch nicht erlangt wurde.

Ergänzende Aspekte der Modelle zur holistischen Bildbetrachtung

Sheridan und Reingold (2017) ist es wichtig zu betonen, dass die zuvor beschriebenen holistischen Ansätze nicht domänenspezifisch entwickelt wurden und auch mit Erkenntnisse aus anderen Forschungsbereichen ergänzt beziehungsweise auch in diesen angewendet werden können. Sie verweisen dabei auf die Arbeiten anderer Wissenschaftlerinnen und Wissenschaftler, die sich ebenfalls mit der visuellen Wahrnehmung beschäftigen. Als interessant gestalten sich sich dabei Studien, die sich mit der visuellen Wahrnehmung in alltäglichen Situationen beschäftigen (Drew, Evans, Võ, Jacobson & Wolfe, 2013; Torralba, Oliva, Castelhano & Henderson, 2006; Wolfe, Võ, Evans & Greene, 2011). Die dabei erlangten Ergebnisse deuten darauf hin, dass Menschen über zwei verschiedene visuelle Verarbeitungskanäle verfügen. Der *non-selective pathway* erlaubt es Menschen auf breitem Feld schnell statistische oder allgemeine visuelle Informationen zu erfassen. Der *selective visual pathway* hingegen ermöglicht eine detailliertere Informationsaufnahme, welche die Erkennung von Objekten unterstützt.

Bei den Theorien von Drew at al. (2013), Torralba et al. (2006) und Wolfe et al. (2011) zeigt sich auf der einen Seite eine Gemeinsamkeit mit der Arbeit von Kundel et al. (2007). Dabei handelt es sich um die parallele Verwendung von zwei Informationsaufnahmeprozessen. Andererseits ergibt sich zu Swensson (1980) jedoch auf der anderen Seite ein Widerspruch, der davon ausgeht, dass die beiden Prozesse nacheinander ablaufen und nicht parallel angewandt werden können.

Gemeinsamkeiten holistischer Ansätze der Bildbetrachtung

Die zuvor von Sheridan und Reingold (2017) vorgestellten Ansätze zur holistischen Betrachtung visueller Stimuli weisen mehrere Gemeinsamkeiten auf. Diese wurden bereits von Kok (2016b) in ihrer Dissertation zusammengefasst.

Die deutlichste Gemeinsamkeit der vorgestellten Modelle stellt die effizientere Suchstrategie von Experten im Vergleich zu Novizen dar. Sheridan und Reingold (2017) beschäftigten sich mit dieser näher und gingen detailliert darauf ein, wie sich diese in den Augenbewegungen der Experten manifestiert. Die nachfolgende Tabelle 2.1 stellt in diesem Zusammenhang auf Basis der holistischen Ansätze den Einfluss von Expertise auf verschiedene Eye-Tracking-Metriken dar und erläutert, in welchen Studien die entsprechenden Beobachtungen gemacht wurden (Sheridan & Reingold, 2017, S.5):

Tabelle 2.1: Metriken der *holistic models of image perception*

Abhängige Variable	Einfluss von Expertise und entsprechende Quellen
Total viewing times	Mit steigender Expertise wird für die Verarbeitung eines jeden Bildes weniger Zeit aufgewendet (Alzubaidi et al., 2009; Assaf et al., 2016; Brunyé et al., 2014; Giovinco et al., 2015; Kok et al., 2016; Krupinski, 1996a,b; Krupinski et al., 2006; Manning et al., 2003; Manning et al., 2006; Wood et al., 2013)
Number of saccades/fixations	Es treten weniger fixations und saccades auf, wenn die Expertise ansteigt (Alzubaidi et al., 2009; Assaf et al., 2016; Brunyé et al., 2014; Donovon & Litchfield, 2013; Krupinski et al., 2006; Krupinski et al., 2013; Manning et al., 2003; Manning et al., 2006; Voisin et al., 2013;)
Saccade length	Mit steigender Expertise treten längere Saccaden auf (Assaf et al., 2016; Kok et al., 2012; Krupinski et al., 2006,Krupinski et al., 2013; Kundel & Nodine, 2004; Kundel et al., 2007; Manning et al., 2003; Manning et al., 2006)
Scanpath length	Die Länge der scanpaths verringert sich mit dem Anstieg von Expertise (Kundel & La Follette, 1972; Krupinski, 1996; Krupinski, 2000a; Krupinski & Borah, 2006)
Time to first fixation on abnormality (search latency)	Die Zeit, bis die Abnormalität erstmals fixiert wird, sinkt mit ansteigender Expertise (Donovon & Litchfield, 2013; Krupinski, 1996; Krupinski, 2000a; Krupinski, 2005; Nodine et al., 1996a,b; Nodine & Kundel, 1997; Kundel /& Nodine, 2004; Kundel et al., 2007; Wood et al., 2013)
Proportion of time fixating relevant regions	Mit ansteigender Expertise werden relevante Regionen prozentual betrachtet länger fixiert (Donovon & Litchfield, 2013; Wood et al., 2013)
Dwell times	Mit steigender Expertise sinkt die dwell time im Allgemeinen (Donovon & Litchfield, 2013; Krupinski et al., 2013; Voisin et al., 2013), steigt jedoch für Auffälligkeiten an (Krupinski, 1996a,b, 2000b; Nodine et al., 1996a,b, 2002; Kundel & Nodine, 2004; Krupinski, 2005)
Fixation times	Die fixations werden expertisebedingt kürzer (Alzubaidi et al., 2009; Kok et al., 2012; Assaf et al., 2016; Giovinco et al., 2015), gleichzeitig lässt sich mit steigender Expertise aber auch beobachten, dass die fixation rate (Anzahl der fixations pro Sekunde) zunimmt (Giovinco et al., 2015)

2.3.4 Die *information-reduction hypothesis*

Eine weitere im Kontext der visuellen Expertise relevante Theorie stellt die *information-reduction hypothesis* dar, welche ursprünglich 1999 von Haider und Frensch entwickelt und unter anderem in den Arbeiten von Kok (2016b), Gegenfurtner et al., (2011) und Jarodzka et al. (2010) aufgegriffen wird. Die Theorie basiert auf der Annahme, dass Expertinnen und Experten einer bestimmten Domäne über eine optimierte Informationsaufnahme verfügen (Gegenfurtner et al., 2011; Jarodzka, Scheiter, Gerjets & van Gog, 2010; Kok, 2016). Während des Expertiseerwerbs in einer Domäne lernen Personen zunehmend besser zwischen relevanten und irrelevanten Informationen zu unterscheiden. Folglich richtet sich ihre Aufmerksamkeit gezielt auf die für die Bearbeitung einer bestimmten Aufgabe relevanten Informationen. Irrelevante Inhalte

werden dagegen weitestgehend ignoriert (Haider & Frensch, 1996, 1999). Bei Novizen ließ sich dagegen im Kontext der *information-reduction hypothesis* feststellen, dass sich deren Aufmerksamkeit vorrangig auf besonders hervorstechende Informationen richtet, obwohl diese für die Erledigung der eigentlichen Aufgabe irrelevant sind (Canham & Hegarty, 2010; Gegenfurtner et al., 2011; Haider & Frensch, 1996, 1996; Hmelo-Silver, 2004; Jarodzka et al., 2010; Kok, 2016; Lowe, 1999).

Empirische Belege für die Anwendung der *information-reduction hypothesis* finden sich in unterschiedlichen Domänen, wobei die Medizin verstärktes Forschungsinteresse erfährt. Kok (2016b) verweist beispielsweise auf Analysen in der Radiologie, bei denen sich Novizen exzessiv mit der im Magen eines Patienten enthaltenen Luft beschäftigen. Dies mag in den gezeigten Aufnahmen zwar äußerst auffällig sein, ist jedoch keine Anomalie und für die Diagnose in diesem Fall irrelevant. Gleichzeitig zeigt sie auf, dass entsprechende Bereiche von Experten ignoriert werden. Ähnliches lässt sich auch in einer Arbeit von Rubin et al. (2015) beobachten, bei welcher es um CT-Aufnahmen der Lunge ging. Es konnte gezeigt werden, dass Experten bei ihrer Betrachtung nur 26% des Lungengewebes scannten, dabei aber 75% der bösartigen Lungenknoten erfassten, welche im Kontext dieser Aufgabe relevant waren. Gesundes Gewebe wurde dagegen in den meisten Fällen ignoriert. Ebenfalls nennenswerte Beispiele aus der Domäne der Medizin finden sich in den Arbeiten von Gegenfurtner et al. (2017), Fox und Falkner-Jones (2017), sowie bei Wolff, Jarodzka und Boshuizen (2017).

Weitere Anwendungsbeispiele für die *information-reduction hypothesis* außerhalb der Domäne der Medizin, lassen sich in den Arbeiten von Antes und Kristjanson (1991), sowie bei Charness Reingold, Pomplun und Stampe (2001) finden. Das erstgenannte Autorenpaar (Antes & Kristjanson, 1991) stellt in einem Experten/Novizen-Vergleich fest, dass erfahrene Künstler eine höhere Dichte an Fixations in den relevanten Arealen eines Bildes aufweisen, als beispielsweise Novizen. Ähnliches lässt sich auch in einer Studie von Charness et al. (2001) beobachten. Diese Forschergruppe beschäftigte sich mit Expertise beim Schach und stellte fest, dass der größte Teil an Fixations den in der jeweiligen Spielsituation relevanten Bereichen des Spielbretts zuteil wird.

Bezüglich einer möglichst allgemeingültigen Operationalisierung der *information-reduction hypothesis* findet sich bei Gegenfurtner et al. (2011) ein Anhaltspunkt. Im Rahmen ihrer Metastudie fassen Gegenfurtner et al. (2011, S.525) die Annahmen der *information-reduction hypothesis* folgendermaßen zusammen: *"If the assumption is true that experts select the amount of information they attend to, then it follows that experts should have fewer fixations of shorter duration on task-redundant areas and more fixations of longer duration on task-relevant areas."* Basierend auf dieser Aussage ergibt sich eine zunehmende Bedeutung für Eye-Tracking-Metriken, welche sich mit Aufmerksamkeit, der Verteilung von Aufmerksamkeit, sowie der Bedeutung von relevanten Arealen beschäftigt. Aufgrund des breiten Anwendungsspektrums, ihres domänenneutralen Hintergrundes und der schwerpunktmäßigen Fokussierung auf Aufmerksamkeit ist

auch im Falle der *information-reduction hypothesis* eine Übertragung auf die Domäne des Software Engineerings möglich.

2.3.5 Die *theory of long-term working memory*

Einen weiteren Erklärungsansatz für visuelle Expertise sehen Gegenfurtner et al. (2011) und Kok (2016b) in der 1995 von Ericsson und Kintsch vorgestellten *theory of long-term working memory*. Diese beschäftigt sich vorrangig mit der Veränderung von Gedächtnisstrukturen bei Experten. Die *theory of long-term working memory* basiert auf der Annahme, dass sich mit zunehmenden Expertisegrad die Informationsverarbeitungskapazität der jeweiligen Person vergrößert. So wird davon ausgegangen, dass Experten neue Informationen schnell aufnehmen, verarbeiten und direkt in ihrem Langzeitgedächtnis ablegen können, welches im Vergleich zu einem Novizen einen hohen Strukturiertheitsgrad aufweist. Dieser hohe Grad an Strukturiertheit ermöglicht es auch, die dort gespeicherten Informationen im Bedarfsfall schnell und effizient abzurufen, wodurch diese auch problemlos auf neue Fälle übertragen werden können (Ericsson & Kintsch, 1995; Gegenfurtner et al., 2011; Kintsch, Patel & Ericsson, 1999; Kok, 2016; van Gog, Ericsson, Rikers & Paas, 2005). Die *theory of long-term working memory* geht de facto davon aus, dass Experten in ihrer jeweiligen Domäne dazu in der Lage sind, die von Cowan (2000) und Miller (1956) postulierten Einschränkungen des Kurzzeit- oder Arbeitsgedächtnisses zu überwinden. In ihrem Fall wird das Langzeitgedächtnis zu einem Langzeitarbeitsgedächtnis, welches bei der Bearbeitung von domänenspezifischen Aufgaben eingesetzt wird (Gegenfurtner et al., 2011; van Gog et al., 2005).

Hinsichtlich einer Übertagung der *theory of long-term working memory* auf die Erforschung von visueller Expertise, sehen Gegenfurtner et al. (2011) vor allem einen Ansatz bezüglich der Metriken, welche mit der Informationsaufnahme und -verarbeitung assoziiert werden (vorrangig *fixation*-basierte und *AOI*-basierte Metriken): *"If we assume that eye movements reflect the processes underlying task performance [...] and if it is true that experts encode and retrieve information more rapidly than novices, then it follows that experts' rapid information processing should be reflected in shorter fixation durations."* (Gegenfurtner et al., 2011, S.524). Diesem Zitat liegt die Annahme von Just und Carpenter (1984) zugrunde, dass sich in den Augenbewegungen die Prozesse widerspiegeln, welche mit der Erfüllung einer bestimmten Aufgabe verbunden sind. Insofern ist davon auszugehen, dass sich das Blickverhalten von Experten im Vergleich zu Novizen dadurch definiert, dass sie über eine deutlich kürzere Fixationsdauer verfügen (Gegenfurtner et al., 2011; Adam Just & Carpenter, 1984).

Ein Bezug zu Code Reviews oder dem Software Engineering lässt sich hier herstellen, wenn man sich mit der Identifikation von Fehlern in einem Quellcode beschäftigt. Experten verfügen aufgrund ihrer Erfahrung über ein reichhaltigeres Wissen über Fehler, Coding Guidelines oder sensible Stellen im Programm.

2.3.6 Die *cognitive load theory*

Die *cognitive load theory* fokussiert im Kontext dieser Dissertation die zur Durchführung des Code Reviews zur Verfügung stehende kognitive Verarbeitungskapazität. Sweller et al. (1998) unterteilen diese in ihrer Theorie in drei sogenannte *loads*. Eine schematische Darstellung der einzelnen *loads* kann Abbildung 2.7 entnommen werden. Diese stellt eine Situation dar, in welcher ein Gleichgewicht zwischen allen drei *loads* herrscht. Sweller et al. al (1998) definieren die *loads* folgendermaßen:

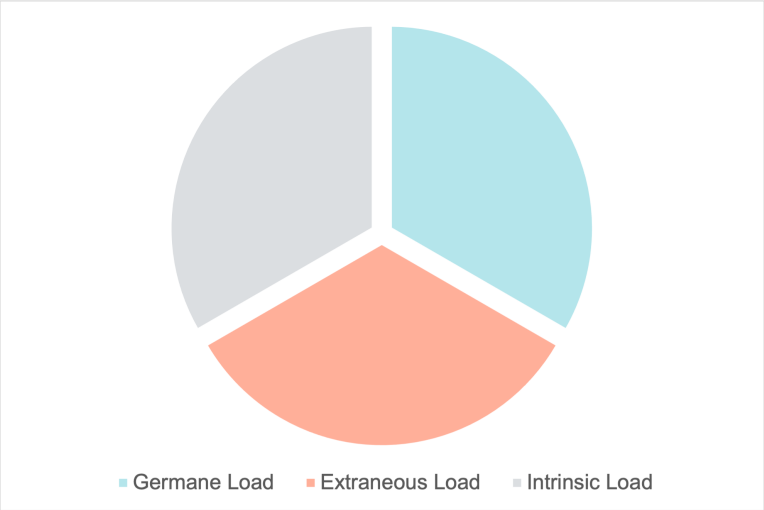

Abbildung 2.7: Schematische Aufteilung der verschiedenen *loads* im Rahmen der *cognitive load theory* (Sweller et al., 1998)

- *Extraneous cognitive load:* Dieser *load* wird durch äußere Störeinflüsse verursacht. So kann sich beispielsweise eine laute und aufgeregte Lern- oder Arbeitsumgebung negativ auswirken und dazu beitragen, dass die ausführende Person abgelenkt wird und folglich nicht ihre volle Leistung erbringen kann. Aus Sicht des Instructional Designs (Sweller et al., 1998) ist es Ziel, den *extraneous cognitive load* durch geeignete Maßnahmen zu verringern, sodass die dabei frei werdenden Kapazitäten auf die eigentliche Erledigung der Arbeit verwendet werden können.

- *Intrinsic cognitive load:* Bei diesem *load* handelt es sich um die Komplexität des zu bearbeitenden Materials oder der zu bearbeitenden Aufgabe. Er wird dadurch verursacht, dass kognitive Kapazitäten benötigt werden, um beispielsweise bestimmte Elemente der Aufgabe zu verstehen und zu lösen. Aus Sicht des Instructional Designs lässt sich dieser nicht verringern, da die Komplexität der Aufgabe feststeht und nicht verändert werden kann (Sweller, 1988). Es bleibt jedoch zu beachten, dass bestimmte Handlungen im Rahmen des Expertiseerwerbs schneller und effizienter ausgeführt werden können, da entsprechende Heuristiken und Automatismen ausgebildet und angewandt werden (Ericsson, 2016; Ericsson & Towne, 2010; Hauser, Reuter, Gegenfurtner, Gruber & Mottok, 2019; Hutzler, Hauser, Reuter, Mottok & Gruber, 2018; Simon & Chase, 1973).

- *Germane cognitive load:* Beim *germane cognitive load* handelt es sich um den *cognitive load* der tatsächlich in die Bearbeitung einer bestimmten Lösung investiert wird. Ziel des Instructional Designs ist es immer, diesen zu maximieren und den *extraneous cognitive load* zu verringern, sodass ein maximaler Output erzielt werden kann (Sweller et al., 1998).

In Bezug auf die hier untersuchten Code Reviews und visuelle Expertise bietet die *cognitive load theory* erweiterte Interpretationsmöglichkeiten hinsichtlich der Komplexität der verwendeten Stimuli. So stellt ich das Wechselspiel zwischen dem *intrinsic* und *germane cognitive load* (Sweller et al., 1998) hinsichtlich eventueller Unterschiede zwischen Experten und Novizen als interessant dar und ermöglicht Rückschlüsse darauf, wie sich domänenspezifische Erfahrung auf die Durchführung der Reviews ausübt.

2.4 Die Physiologie des menschlichen Auges und Eye-Tracking als Erhebungsmethode in der Forschung

Das nachfolgende Unterkapitel beschäftigt sich mit der visuellen Wahrnehmung des Menschen und der dafür relevanten Physiologie, sowie mit dem Einsatz von Eye-Tracking als Erhebungsmethode in der Pädagogik, Psychologie, sowie im Software Engineering. Bezüglich der visuellen Wahrnehmung soll der Aufbau des menschlichen Auges, sowie das visuelle System kurz erklärt werden. Darauf aufbauend wird der Einsatz von Eye-Tracking beschrieben.

2.4.1 Visuelle Wahrnehmung

Bei der Sehfähigkeit handelt es sich nicht nur um die am höchsten entwickelte und komplexeste Sinneswahrnehmung des Menschen. Aufgrund ihrer enormen Bedeutung für das Alltagsleben, stellt sie bezüglich der kognitiven Fähigkeiten auch den meistuntersuchten Bereich dar (Gerrig, 2018). In Bezug auf die visuelle Wahrnehmung des Menschen gilt es zu beachten, dass es sich bei dieser um ein Zusammenspiel zwischen den Augen und dem visuellen System handelt (Duchowski, 2017; Gerrig, 2018; Holmqvist et al., 2011). Dieses Zusammenspiel, sowie die Physiologie des menschlichen Auges soll nachfolgend kurz dargestellt werden.

Die Physiologie des menschlichen Auges

"Often called "the world's worst camera" [...]", so beschreibt Duchowski (2017, S.18) das menschliche Auge. Auch wenn der Vergleich mit einer Kamera nach heutigen Vorstellungen nicht mehr als zeitgemäß erscheint (Leschnik, 2020), eignet er sich dennoch dazu, um die Physiologie und Funktionalität des Auges schematisch darzustellen (Gerrig, 2018). Ein Querschnitt mit allen relevanten Bestandteilen findet sich in der von Gerrig (2018, S.136) übernommenen Abbildung 2.8.

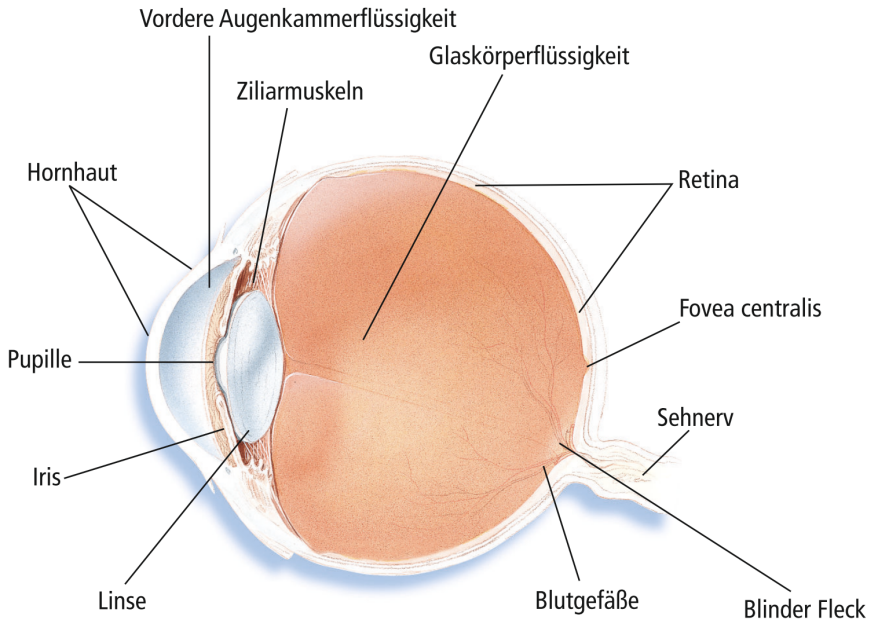

Abbildung 2.8: Die Struktur des menschlichen Auges (übernommen von Gerrig, 2018, S.136)

Im Kontext der vorliegenden Arbeit sind nicht alle Bestandteile des menschlichen Auges relevant, weshalb nachfolgend nur auf die aus Sicht des Eye-Trackings relevanten Bereiche eingegangen wird:

- *Hornhaut:* Bei der Hornhaut handelt es sich um das vorderste, transparente Element des Auges, welches über den Tränenfilm auch in Kontakt mit der Umgebungsluft steht. Aufgrund ihrer exponierten Lage wird ihr auch eine gewisse Schutzfunktion zuteil, weshalb sie im Vergleich zu einer Kamera am ehesten mit der Frontvergütung eines Objektivs oder einem schützenden UV-Filter verglichen werden kann (Bergua, 2017).

- *Iris:* Die Iris, auch als Regenbogenhaut bekannt, besteht aus zwei entgegengesetzten (antagonischen) Muskeln. Durch deren Zusammenspiel kann der Durchmesser der Pupille reguliert werden (Bergua, 2017; Gerrig, 2018). Auf das Beispiel der Kamera übertragen, würde der Iris die Funktion der Blendeneinstellung zuteil werden.

- *Pupille:* Die Pupille ist die zentrale Öffnung des Auges. Sie reguliert, wie viel Licht auf die Retina trifft. Übertragen auf die Vorstellung der Kamera, übernimmt die Pupille die Funktion einer Blende (Bergua, 2017).

- *Linse:* Das durch die Pupille einfallende Licht wird durch die Linse fokussiert, sodass auf der Retina eine scharfe Abbildung entsteht (Duchowski, 2017; Bergua, 2017). Durch Akkommodation kann die Krümmung der Linse so beeinflusst werden, dass entweder nahe oder entfernte Objekte scharf dargestellt werden. Zu beachten ist dabei, dass die Linse die Abbildung auf der Retina auf dem Kopf stehend und spiegelverkehrt darstellt wird (Gerrig, 2018).

- *Retina:* Bei der Retina (oder auch Netzhaut genannt) handelt es sich um den sensorischen Anteil des Auges. Sie befindet sich im hinteren Bereich des Glaskörpers. Ihre Aufgabe ist die Umwandlung von optischen Informationen in neurale Impulse, welche über den Sehnerv an das Gehirn weitergeleitet und dort verarbeitet werden (Bergua, 2017; Duchowski, 2017; Gerrig, 2018). Auf das Beispiel der Kamera übertragen könnte man die Retina am ehesten mit einem Film oder einem digitalen Sensor vergleichen.

- *Fovea centralis:* Die Fovea centralis stellt eine kleine Region in der Nähe des Zentrums der Retina dar. Sie stellt die Stelle des schärfsten Sehens dar. Hier werden Farben und räumliche Details bestmöglich erkannt (Gerrig, 2018). Am Beispiel der Kamera erklärt, müsste man sich hier eher in ein aufgenommenes Foto versetzen: Die Fovea centralis wäre der Bereich des Bildes, auf dem der Fokus liegt.

- *Sehnerv:* Die Weiterleitung der neuralen Impulse an das Gehirn erfolgt über den Sehnerv (Bergua, 2017; Duchowski, 2017; Gerrig, 2018). Bemerkenswert ist, dass die Stelle, an der der Sehnerv mit der Retina verbunden ist keinerlei Rezeptorzellen aufweist. Daher wird dieser Bereich auch als "blinder Fleck" bezeichnet (Gerrig, 2018). Vergleicht man an dieser Stelle wieder mit einer Kamera, so würde es eher Sinn machen, sich an einer Digitalkamera zu orientieren. Der Sehnerv würde hier am ehesten einem USB-Kabel zur Datenübertragung gleichen.

Das visuelle System

Wie bereits einleitend angesprochen wurde, involviert die visuelle Wahrnehmung des Menschen sowohl die Augen, als auch das Gehirn. Die von den Augen wahrgenommenen Reize werden über die Sehnerven in das Gehirn weitergeleitet. Die Sehnerven kreuzen sich im optischen Chiasma. Dabei durchlaufen sie unter anderem den Thalamus (im speziellen den Nukleus), welcher die Informationen zum visuellen Kortex weiterleitet (Duchowski, 2017; Gerrig, 2018). In Bezug auf die relevanten Areale des Gehirns ist der visuelle Kortex von essentieller Bedeutung. Dieser befindet sich am hinteren Ende des Gehirns (Duchowski, 2017; Gerrig, 2018)

Die im Kontext dieses Zusammenspiels relevanten Hirnareale sollen anhand der von Gerrig (2018, S.140) übernommenen Abbildung 2.9 erläutert werden.

2.4.2 Eye-Tracking: Der Begriff und die Technologie

Bei Eye-Tracking handelt es sich um eine Methode, die es Forscherinnen und Forschern erlaubt, die visuelle Aufmerksamkeit von Versuchspersonen in bestimmten Situationen zu untersuchen und zu verstehen. Nachfolgend soll kurz darauf eingegangen werden, wie sich Eye-Tracking entwickelt hat, wie moderne Geräte arbeiten und welche Metriken von diesen produziert und von der Forschung verwendet werden.

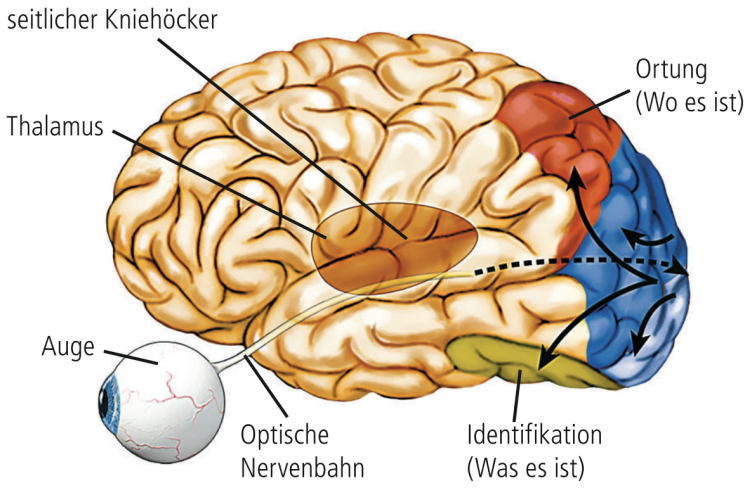

Abbildung 2.9: Sehbahnen des menschlichen visuellen Systems (übernommen von Gerrig, 2018, S.140)

Historische Entwicklung des Eye-Trackings

Beim Begriff Eye-Tracking werden in der heutigen Zeit häufig Assoziationen mit Brillen geweckt, die die Blickrichtungen von Personen erfassen. Dabei handelt es sich aber nur um die halbe Wahrheit. Diese sogenannten head-mounted Eye-Tracker stellen lediglich einen einzelnen Typ der derzeit geläufigen Geräte dar. Die Ursprünge dieser Technologie führen deutlich weiter zurück (Duchowski, 2017; Holmqvist et al., 2011; Lemahieu & Wyns, 2010).

Die ersten ernsthaften Versuche, die Blickbewegungen eines Menschen aufzuzeichnen lassen sich im späten 19. Jahrhundert finden (Delabarre, 1898; Haselhuhn, 2020; Holmqvist et al., 2011; Huey, 1898). Dabei handelt es sich jedoch um "Technologie" die den jeweiligen Probanden einiges abverlangte. Bergstrom und Schall (2014) sprechen in diesem Kontext scherzhaft von "mittelalterlichen Folterwerkzeugen" (S.14). Besonders erwähnenswert sind diesbezüglich die Ansätze von Huey (1898) und Delabarre (1898). Bei beiden Vorgehensweisen wurden den Versuchspersonen die jeweiligen Messinstrumente in Form von Kontaktlinsen direkt am Augapfel befestigt (Delabarre, 1898; Haselhuhn, 2020; Huey, 1898). Diese Kontaktlinsen waren über Federn mit einem Stift verbunden. Je nachdem, wie die Probanden ihre Augen bewegten, zeichnete der Stift entsprechend mit (Delabarre, 1898; Holmqvist et al., 2011; Huey, 1898).

Unterschiede zwischen Huey (1898) und Delabarre (1898) zeigen sich im von Ihnen verwendeten Material ihrer Kontaktlinsen. Im Fall von Huey (1898) bestanden diese aus Gips, wohingegen Delabarre (1898) Metall verwendete. Gemein haben beide Verfahren, dass man sie sich als äußerst schmerzhaft vorstellen kann, weshalb sie so de facto nie für größere Studien eingesetzt werden konnte (Bergstrom & Schall, 2014). So ist beispielsweise bekannt, dass Delabarre (1898) seine Augen mit mehreren Tropfen Kokain betäuben musste, um seine Konstruktion überhaupt testen zu können. Eine beispielhafte, wenn auch modernere Darstellung von Delabarres Kontaktlinsen findet sich in Abbildung 2.10:

Abbildung 2.10: Beispielhafte Darstellung von Delabarres (1898) Kontaktlinse (übernommen von Chronos-Vision (2022))

Einen langfristigen und nachhaltigen Beitrag zur Wissenschaft konnten die Arbeiten von Delabarre (1898) und Huey (1898) durch die Notationsformen der Augenbewegungen erzielen, die von ihren Kontaktlinsen mitgezeichnet wurden (Haselhuhn, 2020). Bei diesen Verfahren wurden Fixationen als Punkte (oder punktähnliche Abbildungen) und Sakkaden als Linien dargestellt. Diese Darstellungsform der Augenbewegungen wird auch heute noch herstellerübergreifend von entsprechender Experimentalsoftware verwendet und kann als Standard in der Eye-Tracking-Forschung betrachtet werden (Duchowski, 2017; Haselhuhn, 2020; Holmqvist et al., 2011; SR Research, 2022; Tobii Pro, 2021).

Mit dem Beginn des 20. Jahrhunderts, bereits wenige Jahre nach den Arbeiten von Huey (1898) und Delabarre (1898), wurde ein deutlich weniger invasives Verfahren entwickelt. Dieses wurde 1901 von Dodge und Cline vorgestellt. Dabei werden die Lichtspiegelungen einer externen Lichtquelle auf der Cornea einer Versuchsperson abfotografiert. Dodge und Cline (1901) arbeiteten vor mehr als hundert Jahren zwar mit Kameratechnik, die noch in den Kinderschuhen steckte, die dahinterstehende Idee, die sie dabei prägten, stellt jedoch die Grundlage der modernen Eye-Tracking-Technologie dar (Duchowski, 2017; Holmqvist et al., 2011; Holmqvist & Andersson, 2017). Eine Verbesserung dieser Idee findet sich in einer 1950 von Fitts, Jones und Milton publizierten Arbeit, die sich mit den Augenbewegungen von Piloten während einer Instrumentenlandung beschäftigt. Die Forscher setzten hier erstmalig eine Filmkamera ein, mit welcher die Augenbewegungen der Versuchspersonen gefilmt wurden.

Besonders erwähnenswert ist in dieser Zeitepoche die Arbeit von Alfred Lukyanovich Yarbus (1967). Dieser gilt als Entdecker des *scanpaths*, welcher eine zielgerichtete serielle Abfolge von *fixations* und *saccades* darstellt (Duchowski, 2017; Haselhuhn, 2020; "Eye movements and vision", 1967). Gleichzeitig liefert Yarbus in seiner 1967 publizierten Arbeit sehr detaillierte Beschreibungen seines Forschungsequipments

und was es bei Arbeiten mit diesem zu beachten gilt. Eine Fotografie (siehe Abbildung 2.11 dieser Apparatur zeigt bereits gewisse Ähnlichkeiten zu den aktuellen Modellen auf.

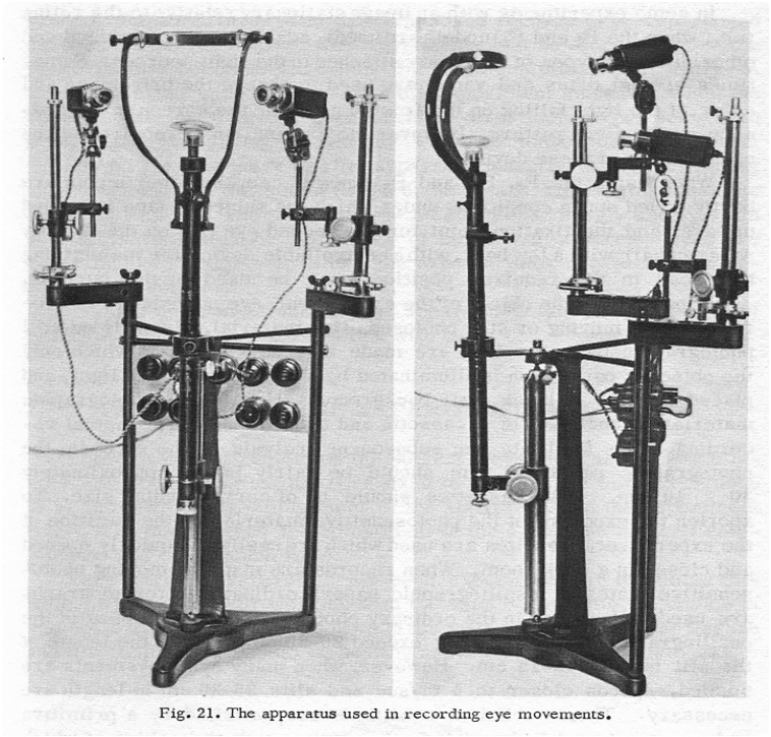

Abbildung 2.11: Prototyp eines analogen Eye-Trackers (Abbildung übernommen von Yarbus (1967, S.41))

Obwohl diese Form des Eye-Trackings bereits deutlich weniger invasiv und genauer ist, als die Arbeiten des späten 19. Jahrhunderts und frühen 20. Jahrhunderts (Delabarre, 1898; Dodge & Cline, 1901; Huey, 1898), ist sie für die Versuchspersonen dennoch mit einigen Unannehmlichkeiten und Einschränkungen verbunden. So müssen beispielsweise die Augenlieder gespreizt und mit Pflasterstreifen fixiert werden. Gleichzeitig ist darüber hinaus in vielen Fällen noch das Tragen einer Plexiglasmaske erforderlich und auch die Verwendung einer Kinnstütze ist notwendig ("Eye movements and vision", 1967).

Durch die voranschreitende Miniaturisierung der erforderlichen Technologien, die Verbreitung von weniger invasiven Verfahren und einer leichteren Handhabbarkeit der Geräte für die Versuchsleiterinnen und -leiter wurde Eye-Tracking ab der Mitte des 20. Jahrhunderts einer immer größeren Forschergruppe zugänglich. Dennoch sollte es bis in die 1970er und 1980er Jahre dauern, bis diese Technologie in größeren Stückzahlen für die Forschung verfügbar wurde (Haselhuhn, 2020; Holmqvist et al., 2011; Józsa & Hámornik, 2012; Lemahieu & Wyns, 2010).

Funktionsprinzip moderner Eye-Tracker

Moderne Eye-Tracker (siehe z.B. Abbildung 2.12), wie sie auch im Rahmen der Studien dieser Dissertation verwendet worden sind, stellen eine Kombination aus

aufeinander abgestimmter Hard- und Software dar. Um für die Versuchspersonen möglichst wenig invasiv zu sein, werden deren Augen bei der Verwendung von aktuellen Systemen von (mindestens) einer Infrarotlichtquelle angeleuchtet. Dieses Lichtspektrum ist für das menschliche Auge nicht wahrnehmbar, wodurch die Versuchsteilnehmerinnen und -teilnehmer auch nicht geblendet werden (Duchowski, 2017).

Abbildung 2.12: Tobii Pro Spectrum und Fusion als Beispiel für einen modernen, kompakten Eye-Tracker (übernommen von Tobii (2020a, 2020b)

Das auf das Auge treffende Infrarotlicht erzeugt (mindestens) zwei Lichtspiegelungen, welche als Fixpunkte dienen (Cornelissen, Peters & Palmer, 2002; Duchowski, 2017; Holmqvist et al., 2011). Bei diesen Fixpunkten handelt es sich um die Pupille, welche aufgrund ihres hohen Schwarzanteils einen sehr guten Kontrast liefert und die Spiegelung auf der Hornhaut, die auch als *corneal reflection* bekannt ist (Duchowski, 2017; Haselhuhn, 2020; Holmqvist & Andersson, 2017; Holmqvist et al., 2011). Die Infrarotbeleuchtung hat auch zu Folge, dass eine sogenannte *bright pupil* erzeugt wird, wie sie beispielsweise in einer Studie von Smith und Graham (2006, S.3) in Abbildung 2.13 dargestellt wird.

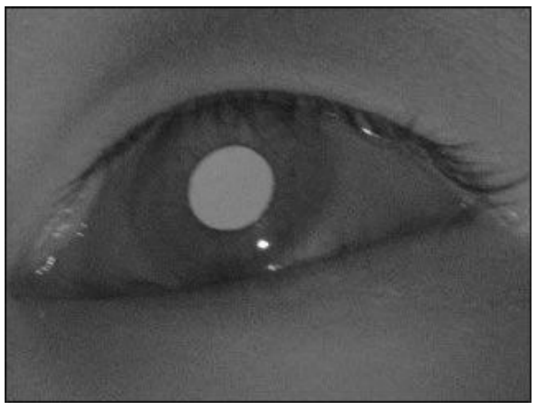

Abbildung 2.13: *Bright Pupil* und *Corneal Reflection* (übernommen von Smith & Graham, 2006, S.3)

An der Abbildung der *bright pupil* (Smith & Graham, 2006, S.3) können auch die genannten Fixpunkte erkannt werden. Diese können von den Infrarotkameras eines modernen Eye-Trackers erfasst und in Bezug auf den gezeigten visuellen Stimulus

von einer entsprechenden Experimentalsoftware berechnet werden. (Cornelissen et al., 2002; Duchowski, 2017; Haselhuhn, 2020; Holmqvist et al., 2011; Sharafi et al., 2020). Als Output ergibt sich ein Datensatz mit entsprechenden Daten, die Rückschlüsse über die Augenbewegungen der Versuchspersonen ermöglichen.

Kapitel 3

Aktueller Forschungsstand zu visueller Expertise bei Code Reviews

Dieses Kapitel erschließt den aktuellen Forschungsstand zur visuellen Expertise bei Code Reviews und fasst diesen in der Form eines systematischen Literaturvergleichs zusammen.

Aufgrund der Breite und den verschiedenen Facetten des behandelten Themas werden dabei nur Arbeiten berücksichtigt, welche einen direkten Bezug zum Dissertationsvorhaben aufweisen. Der Fokus liegt somit auf Arbeiten, die von den Programmiererinnen und Programmierern eine direkte Auseinandersetzung mit Quellcode erfordern und deren Augenbewegungen mittels Eye-Tracking aufgezeichnet und analysiert werden.

3.1 Erschließung des aktuellen Forschungsstandes

Die Erschließung des aktuellen Forschungsstandes dieser Arbeit basiert auf zwei Säulen: Bereits publizierte Metastudien (Obaidellah et al., 2018; Sharafi et al., 2015) und ein eigens erstellter systematischer Literaturvergleich.

Bezüglich der Metastudien wird auf die Arbeiten von Sharafi et al. (2015) und Obaidellah et al. (2018) zurückgegriffen. Der erstgenannten Publikation kommt aufgrund ihrer früheren Verfügbarkeit eine große Bedeutung zu, da sie die für die Anfertigung dieser Dissertation relevanten Rechercheprozesse maßgeblich beeinflusst hat und gleichzeitig mit ihren Erkenntnissen ein tragendes Element im theoretischen Grundkonzept darstellt (Sharafi et al., 2015). Die Ergebnisse, das methodische Vorgehen, sowie die für die Erstellung verwendeten Studien beider Arbeiten wurden überprüft und für die Anfertigung eines eigenen systematischen Literaturvergleichs genutzt.

Der eigene systematische Literaturvergleich deckt den Zeitraum zwischen 2014 und 2020 ab. Diese Zeitspanne stellt den für das Design der eigenständig durchge-

führten Arbeiten (siehe Kapitel 5 und Kapitel 6) relevanten Zeitraum dar. Thematisch beschäftigt sich der systematische Literaturvergleich mit der Schnittmenge von Eye-Tracking und Code Comprehension bzw. Debugging.

Sowohl für die Metastudien von Sharafi et al. (2015) und Obaidellah et al. (2018), als auch für den selbst erstellten systematischen Literaturvergleich gilt eine zentrale Anforderung, welche zum Ausschluss mehrerer Studien geführt hat: Die Versuchspersonen der analysierten Studien müssen im Kontext der durchgeführten Experimente direkt mit Quellcode interagiert haben und während dieser Interaktion müssen die Augenbewegungen mittels eines Eye-Trackers erfasst und aufgezeichnet worden sein.

Nachfolgend soll auf die beiden bereits existierenden Literaturvergleiche von Sharafi et al. (2015) (siehe Absatz 3.1.1 und Obaidellah et al. (2018) (siehe Absatz 3.1.2) eingegangen werden, bevor geschildert wird, wie der Forschungsstand durch eine eigenständige Recherche erweitert wird (siehe Absatz 3.2.

3.1.1 Der Literaturvergleich von Sharafi, Soh und Guéhéneuc (2015)

Die Metastudie von Sharafi, Soh und Guéhéneuc, wurde 2015 unter dem Titel *"A systematic literature review on the usage of eye tracking in software enginnering"* in der Zeitschrift *"Information and Software Technology"* veröffentlicht. Bei dieser Arbeit handelt es sich um einen systematischen Literaturvergleich (Kitchenham et al., 2009). Dieser greift auf die in der Informatik und anderen Domänen verbreitete Methodik von Kitchenham et al. (2009) zurück. Im Rahmen dieser Metastudie (Sharafi et al., 2015) werden alle thematisch relevanten Studien, welche zwischen 1990 und 2014 veröffentlicht wurden erfasst, analysiert und zusammengefasst.

Neben der Zusammenfassung des Forschungsstandes leisten Sharafi et al. (2015) einen großen Beitrag zur Strukturierung des Themenfeldes. Die Autorinnen und Autoren unterteilen auf Basis der verwendeten Stimuli und der auszuführenden Aufgaben die analysierten Studien in fünf verschiedene Kategorien. Bei diesen handelt es sich um *code comprehension*, *debugging*, *model comprehension*, *collaborative interatcions* und *traceability*. Der thematische Schwerpunkt des hier vorliegenden Dissertationsvorhabens liegt auf Code Reviews, welche im Literaturvergleich von Sharafi et al. (2015) durch die Themen *code comprehension* und *debugging* bestmöglich abgebildet werden. Dies ist darauf zurückzuführen, dass Programmiererinnen und Programmierer in diesen beiden Bereichen die intensivste Interaktion mit dem Code durchleben. Weiterhin stellen beide Teilbereiche essentielle Bestandteile bei der Realisierung und Qualitätssicherung von umfangreichen Softwareprojekten dar (Hoffmann, 2013; Sommerville, 2012; Spillner & Linz, 2005). Im Kontext des Literaturvergleichs von Sharafi et al. (2015) sind die beiden Themenbereiche folgendermaßen zu verstehen:

- *Code comprehension*, das Verstehen von Quellcode, kann als die zentrale Fähigkeit eines Programmierers gesehen werden (Busjahn, Bednarik & Schulte, 2014). Darunter versteht man die Fähigkeit, einen Code zu lesen und diesen in seiner

vollen Bedeutung auch zu verstehen. Ohne *code comprehension* sind auch keine weiterführenden Aufgaben oder beispielsweise die Realisierung von großen Softwareprojekten möglich (Bednarik, Schulte, Budde, Heinemann & Vrzakova, 2018; Busjahn, Bednarik & Schulte, 2014; Busjahn et al., 2015; Czerwonka et al., 2015). Diese Fähigkeit wird umso bedeutender, wenn man sich die aktuellen Entwicklungen in modernen Softwareunternehmen vergegenwärtigt (Busjahn, Bednarik & Schulte, 2014; Kononenko et al., 2016): In der Industrie ist es üblich, dass große Softwareprojekte zwischen mehreren Programmiererinnen und Programmierern an verschiedenen Standorten aufgeteilt, dezentral entwickelt und abschließend zu einer fertigen Version zusammengefügt werden. Da manche Betriebe mit einer hohen Personalfluktuation zu kämpfen haben, muss selbst neues (und verhältnismäßig unerfahrenes) Personal dazu in der Lage sein, sich schnell in den Code einzuarbeiten und sich für diesen ein fundiertes Verständnis anzueignen. Ohne diesem Verständnis sind Aufgaben wie die Wartung, Anpassung oder das Updaten eines Programmes nicht möglich (Ernst et al., 2016; Hoffmann, 2013; Sommerville, 2012).

- Das *Debugging* stellt einen wichtigen Schritt in der Qualitätssicherung eines Softwareprojekts dar (Bacchelli & Bird, 2013; Czerwonka et al., 2015; Edmundson et al., 2013; Sharafi et al., 2015). Spillner und Linz (2005) definieren das *Debugging* als das Lokalisieren und Beheben eines Defekts. Meist ist nur bekannt, wie sich ein Fehler auswirkt, jedoch nicht dessen Position im Code. Durch das Entfernen oder eine entsprechende Modifikation des Codes erhoffen sich die Verantwortlichen eine Qualitätsverbesserung des Programs. Klassischerweise übernimmt ein Softwareentwickler diese Aufgabe und führt entsprechende Maßnahmen zum *Debugging* durch. In der modernen Softwareentwicklung werden die Entwickler dabei häufig auch von speziellen Programmen, sogenannten *Debuggern*, unterstützt (Czerwonka et al., 2015; Edmundson et al., 2013; Lettnin & Winterholer, 2017; Sommerville, 2012). Spillner und Linz (2005) legen auch Wert auf eine eindeutige Abgrenzung zwischen *debugging* und Testen. Beide Begriffe wirken zwar unter Berücksichtigung des Aspekts der Qualitätssicherung sehr ähnlich, unterscheiden sich jedoch grundlegend in der Zielsetzung. Das *Debugging* hat das Ziel, Fehlerzustände zu beheben, wohingegen das Testen darauf abzielt diese Zustände systematisch und gezielt aufzudecken und zu untersuchen.

Sharafi et al. (2015) leisten mit ihrem Literaturvergleich noch einen weiteren Beitrag zur Erforschung von Augenbewegungen im Software Engineering: Aufgrund der Kategorisierung des Themenfeldes, sowie einer detaillierten Schilderung bei der Erstellung ihrer Arbeit ist diese für Forschende in diesem Bereich ohne größere Mühen replizierbar. So listen Sharafi et al. (2015,S.82-83) beispielsweise ihre verwendeten Suchterme und die für die Recherche genutzten Plattformen auf und machen diese somit der Forschungsgemeinschaft zugänglich.

Insgesamt identifizieren Sharafi et al. (2015) 36 Studien zum Thema Eye-Tracking im Software Engineering und werden diese in ihrer Arbeit aus. Für das hier angestrebte Dissertationsvorhaben sind davon insgesamt 21 relevant. Bei einer eigenen Literaturrecherche, basierend auf den von Sharafi et al. (2015) genannten Grundlagen konnte auf insgesamt 20 der ausgewählten Paper Zugriff erlangt werden. Der Prozess der eigenen Literaturrecherche soll näher im Absatz 3.2 geschildert werden.

3.1.2 Der Literaturvergleich von Obaidellah, al Haek und Cheng (2018)

Die Arbeit von Sharafi et al. (2015) wird durch eine weitere Metastudie von Obaidellah et al. (2018) erweitert. Die Autorin und ihre Kollegen listen in ihrem systematischen Literaturvergleich *A survey on the useage of eye-tracking in computer programming* (ebenfalls basierend auf der Methode von Kitchenham et al. (2009) insgesamt 63 Studien auf. Im Vergleich zu Sharafi et al. (2015) gehen Obaidellah et al. (2018) speziell auf den Aspekt der Programmierung ein, was einen stärkeren Fokus auf die Interaktion mit Quellcode zur Folge hat. Sie unterteilen die gefundenen Ergebnisse in vier Gruppen. Von Relevanz für das Dissertationsvorhaben sind dabei *Program/Code comprehension* (26 Studien) und *Debugging* (19 Studien). Es gilt zu beachten, dass die Arbeit von Obaidellah et al. (2018) die Ergebnisse von Sharafi et al. (2015) inkludiert. Bereinigt man diese ergeben sich insgesamt 15 neue Publikationen.

3.2 Durchführung eines eigenständigen systematischen Literaturvergleichs

Die Arbeiten von Sharafi et al. (2015) und Obaidellah et al. (2018) stellen eine gute Ausgangsbasis für die Durchführung einer eigenständigen Literaturrecherche dar. Beide Autorengruppen haben in ihren Arbeiten die eigene Vorgehensweise transparent und detailliert dargestellt, weshalb diese zur weiteren Erschließung des Forschungsstandes reproduziert wird.

3.2.1 Vorgehensweise nach Kitchenham et al. (2009)

Aufbauend auf den Erkenntnissen der genannten Metastudien (Obaidellah et al., 2018; Sharafi et al., 2015) und unter Verwendung der von Kitchenham et al. (2009) vorgestellten Methode, wird im Rahmen der hier vorliegenden Dissertation ein eigener Literaturvergleich durchgeführt. Bei dessen Realisierung ergibt sich allerdings ein Problem: Die Metadatenbank *Engineering Village* ist zum aktuellen Zeitpunkt (03.07.2019) weder von der OTH Regensburg, noch der Universität Regensburg lizenziert. Dies steht einer exakten Replikation des Literaturvergleichs von Sharafi et al. (2015) im Wege und zwingt zur Anwendung von alternativen Suchstrategien und einer Neukonzeption des Vergleichs im Sinne der angestrebten Methodik (Kitchenham et al., 2009).

Ein Großteil der in den Literaturvergleichen vorgestellten Studien stammt aus Zeitschriften oder Konferenzbänden der beiden großen Ingenieurs- und Forschungsvereinigungen IEEE (z.B. ICSE, ICSM, ...) und ACM (z.B. ETRA, CSCW, ...) oder werden mit diesen assoziiert (Obaidellah et al., 2018; Sharafi et al., 2015). Beide Vereinigungen verfügen über umfangreiche Datenbanken, in denen alle Publikationen zugänglich sind. Die OTH Regensburg verfügt sowohl für *IEEE Xplore*, als auch die *ACM Library* über vollständige Zugriffsrechte. Sharafi et al. (2015) decken im Literaturvergleich den Zeitraum von 1990 bis 2014 ab, Obaidellah et al. (2018) erweitern diesen bis 2017. Der eigene Literaturvergleich soll den Forschungsstand bis zur Durchführung der letzten Studie dieser Dissertation (siehe Kapitel 6) abdecken und gleichzeitig eventuell vorhandene Lücken zwischen den beiden genannten Arbeiten (Obaidellah et al., 2018; Sharafi et al., 2015) schließen. Somit adressiert die eigene Literaturrecherche den Zeitraum von 2014 bis 2019. Wesentliches Suchkriterium des Vergleichs sind die von den Autoren der einzelnen Studien vergebenen Keywords. Diese müssen in Bezug auf die Belange der eigenen Arbeit entsprechend ausgewählt und angepasst werden. Da es sich bei der Kombination aus Software Engineering und Eye-Tracking noch immer um eine verhältnismäßig kleine Schnittmenge handelt (insbesondere durch die thematische Fokussierung auf eine Interaktion mit Quellcode), wird auf die Anwendung von Suchtermen wie bei Sharafi et al. (2015, S.83) verzichtet. Stattdessen wird in den relevanten Datenbanken mit einzelnen Begriffspaaren gesucht. Diese Vorgehensweise geht zwar mit einem größeren Arbeitsaufwand und einer länger andauernden Selektion der erhaltenen Studien einher, dadurch wird jedoch vermieden, dass ansonsten nicht auffindbare Paper nicht erfasst werden. Konkret ergeben sich aus den Anforderungen der eigenen Arbeit und dem Literaturvergleich von Sharafi et al. (2015) folgende fünf Keyword-Paare:

1. *Software Engineering* AND *Eye Tracking*
2. *Code Reviews AND Eye Tracking*
3. *Programming AND Eye Tracking*
4. *Debugging AND Eye Tracking*
5. *Code Comprehension* AND *Eye Tracking*

Mit den oben genannten Keywords werden die Datenbanken von IEEE und ACM nach relevanten Arbeiten durchsucht. Diese werden anhand des Titels und des Abstracts hinsichtlich ihrer Eignung für die hier praktizierte Forschung untersucht und im Anschluss daran heruntergeladen und genauer geprüft. Als zusätzliche Validierungsmaßnahme wird eine Autorensuche durchgeführt, bei der die Publikationen aller involvierten Wissenschaftlerinnen und Wissenschaftler hinsichtlich weiterer Treffer überprüft werden.

3.2.2 Ergebnisse der Datenbankabfrage

Die Abfrage der Datenbanken mit den jeweiligen Keywords liefert für *IEEE Xplore* 340 und für die *ACM Library* 184 Treffer. Daraus resultieren insgesamt 511 Treffer. Es ist allerdings zu beachten, dass diese Resultate noch nicht gefiltert sind und somit kein relevanter Bezug zum Thema der Dissertation vorliegen muss. Nachfolgend sollen die beiden einzelnen Abfragen und die jeweils erhaltenen Ergebnisse kurz erläutert werden.

IEEE Xplore

Bei der Datenbank *IEEE Xplore* handelt es sich um die digitale Bibliothek der IEEE. Zum Ende der eigenen Literaturrecherche (08.07.2019) umfasste sie insgesamt 4.897.427 Publikationen. Mit wissenschaftlichen Veröffentlichungen, Zeitschriften, sowie international gültigen Standards und Regeln handelt es sich bei ihr um ein äußerst umfangreiches und weit verbreitetes Tool, welches jedem Informatiker, Maschinenbauer oder Physiker bekannt ist. Die Abfrage der relevanten Literatur liefert folgende Ergebnisse:

1. *Software Engineering* AND *Eye Tracking*: 120 Treffer - davon 18 relevant
2. *Code Reviews AND Eye Tracking*: 6 Treffer - davon 1 relevant
3. *Programming AND Eye Tracking*: 178 Treffer - davon 18 relevant
4. *Debugging AND Eye Tracking*: 11 Treffer - davon 7 relevant
5. *Code Comprehension* AND *Eye Tracking*: 25 Treffer - davon 16 relevant

Insgesamt ergeben sich in der Summe mit diesen Keywords 340 Treffer bei *IEEE Xplore*. Nach einer ersten Durchsicht, die lediglich auf den Keywords und dem Titel der jeweiligen Arbeit basiert, kommen 60 Arbeiten in die engere Auswahl für das Dissertationsvorhaben. Die 60 ausgewählten Arbeiten werden in einem zweiten Schritt genauer überprüft. Dabei wird das Abstract gelesen und untersucht, ob es sich bei den Publikationen um empirische Arbeiten handelt, bei denen wirklich ein Eye-Tracker eingesetzt worden ist und ob dabei Originaldaten vorliegen. Wendet man diese Kriterien an, so wird die Auswahl um 44 Publikationen verringert, woraus insgesamt 16 relevante IEEE-Publikationen resultieren.

ACM Library

Auch die *Association for Computing Machinery (ACM)* verfügt mit der *ACM Library* über eine äußerst umfangreiche Datenbank, die durchaus mit der zuvor beschriebenen *IEEE Xplore* mithalten kann. Im Vergleich zu dieser Plattform legt die *ACM Library* jedoch einen stärkeren Fokus auf Publikationen und weniger auf Standards und Normen. Die Datenbank wird ebenfalls mit den bereits erwähnten Suchtermen abgefragt:

1. *Software Engineering* AND *Eye Tracking*: 58 Treffer - davon 26 relevant
2. *Code Reviews AND Eye Tracking*: 6 Treffer - davon 3 relevant
3. *Programming AND Eye Tracking*: 86 Treffer - davon 38 relevant
4. *Debugging AND Eye Tracking*: 7 Treffer - davon 6 relevant
5. *Code Comprehension* AND *Eye Tracking*: 27 Treffer - davon 25 relevant

Bei der Abfrage der *ACM Library* zeigt sich, dass die beiden Organisationen ACM und IEEE sehr eng miteinander verflochten sind. Beide Vereinigungen verbindet eine intensive Kooperation, welche sich in Form von gemeinsam organisierten Konferenzen oder publizierten Zeitschriften widerspiegelt. Dies hat zur Folge, dass es bei der Datenbankabfrage unweigerlich zu Überschneidungen kommt. Trotz einer geringeren Gesamttrefferzahl (184) konnten insgesamt 98 Studien auf Grundlage der Keywords gefunden werden. Bereinigt man nun eventuelle Überschneidungen mit der IEEE-Datenbank, so bleiben 51 neue Studien übrig. Diese werden wieder auf Grundlage ihres Abstracts beurteilt und es wird untersucht ob es sich um Originalstudien handelt, in deren Datenerhebung ein Eye-Tracker eingesetzt worden ist. Durch die Anwendung dieser Kriterien verringert sich die Anzahl der Studien auf 22 verbleibende Arbeiten.

Ausschluss und Bewertung der gesammelten Paper

Die gesammelten Paper werden in einem nächsten Schritt gelesen, bewertet und gefiltert. Studien, bei denen kein Eye-Tracker verwendet worden ist, keine Originaldaten erhoben wurden oder sich auf technische Aspekte (beispielsweise Tools oder Algorithmen) konzentriert wurde, wurden ausgeschlossen. Durch die Anwendung dieser Kriterien lässt sich die Gesamtzahl der Paper auf 24 reduzieren. Diese werden nachfolgend mit den Arbeiten von Sharafi et al. (2015) uns Obaidellah et al. (2018) abgeglichen, wodurch Überschneidungen vermieden und vorhandene Lücken geschlossen werden sollen. Ebenso soll durch dieses Vorgehen ein Gesamtbild erstellt werden, welches letztendlich den finalen Stand der Forschung zu diesem Thema darstellt. Eine Grafische Darstellung der Ergebnisse des eigenen systematischen Literaturrecherche findet sich nachfolgend in Grafik 3.1.

3.2.3 Auflistung der relevanten Studien

Nach einer entsprechenden Datenfilterung und unter Berücksichtigung der Themen des Dissertationsvorhabens gestalten sich insgesamt 40 Studien als relevant. Diese sollen nachfolgend in Tabelle 3.1 aufgeführt und kurz zusammengefasst werden. Aus Sicht der vorliegenden Dissertation gestalten sich die nachfolgend aufgelisteten Studien als relevant:

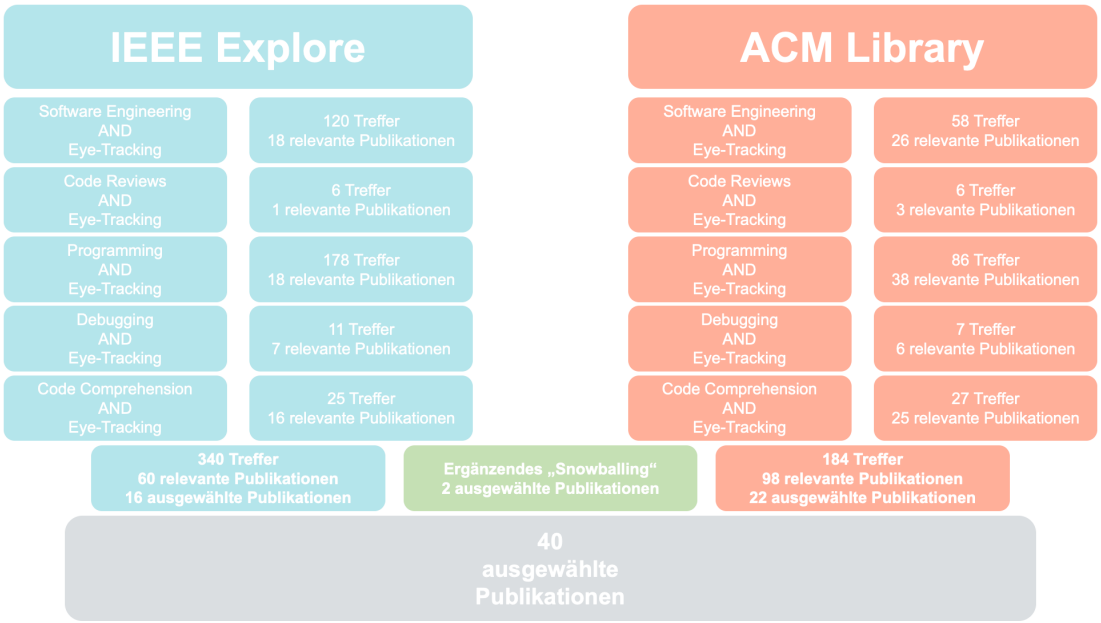

Abbildung 3.1: Grafische Darstellung der Ergebnisse der eigenen systematischen Literaturrecherche

Autoren	Jahr	Titel	Zusammenfassung
Crosby & Stelovsky	1990	How do we read algorithms? A case study	Die Autoren (Crosby & Stelovsky, 1990) untersuchen das Lesen von Quellcode und den Einfluss von Expertise auf die Verstehensstrateigen, sowie die damit verbundenen Augenbewegungen. Sie stellen fest, dass Anfänger mehr visuelle Aufmerksamkeit auf Kommentare verwenden.
Crosby, Scholz & Wiedenbeck	2002	The roles beacons play in comprehension for novice and expert programmers	Die Studie identifiziert Beacons als eine entscheidende Komponente in Bezug auf code comprehension. Weiterhin werden erfahrungsspezifische Unterschiede untersucht (Crosby, Scholtz & Wiedenbeck, 2002).
Aschwanden & Crosby	2006	Code scanning patterns in program comprehension	Die Autoren (Aschwanden & Crosby, 2006) konzentrieren sich auf expertisebedingte Veränderungen bei Verständnisaufgaben. Es zeigt sich, dass Experten und Anfänger unterschiedlich viel visuelle Aufmerksamkeit auf verschiedene Teile des Quellcodes verwenden.
Bednarik & Tukiainen	2006	An eye-tracking methodology for characterizing program comprehension processes	Es wird eine Visualisierungstechnik für Java-Code vorgestellt und mittels Eye-Tracking untersucht (Bednarik & Tukiainen, 2006). Bei dieser ist es möglich, den Code und die Visualisierung abwechselnd zu betrachten.
Uwano, Nakamura, Monden & Matsumoto	2006	Analyzing individual performance of source code review using reviewer's eye movement	Eine Forschergruppe (Uwano, Nakamura, Monden & Matsumoto, 2006) beschäftigt sich mit den Reviews von Quellcode und welche Bedeutung der *scan time* in Bezug auf das Finden von Fehlern zuteil wird.

Autoren	Jahr	Titel	Zusammenfassung
Bednarik & Tukiainen	2007	Validating the restricted focus viewer: A study on using eye movement tracking	Die Forscher (Bednarik & Tukiainen, 2007) nutzen sowohl einen Eye-Tracker, als auch RFV um zu untersuchen, welche Einflüsse RFV und Expertise auf die Aufmerksamkeit der Versuchspersonen hat.
Bednarik & Tukiainen	2008	Temporal eye-tracking data: Evolution of debugging strategies with multiple representations	Es wird eine vertiefte Analyse der 2007er-Studie durchgeführt. Diese konzentriert sich vorrangig auf die Eye-Tracking-Daten. Sie segmentieren diese in kleinere Einheiten und können so Unterschiede zwischen Experten und Novizen genauer betrachten (Bednarik & Tukiainen, 2008).
Sharif & Maletic	2010	An eye-tracking study on camel-case and under score identifier styles	Die Studie (Sharif & Maletic, 2010a) untersucht, welchen Einfluss die Namenskonvention für Identifier auf das Verständnis von Code hat. Obwohl sich in den Augenbewegungen keine signifikanten Genauigkeitsunterschiede zwischen den beiden Stilen feststellen lässt, erkennen die Teilnehmerinnen und Teilnehmer Identifier im underscore-Stil schneller
Busjahn, Schulte & Busjahn	2011	Analysis of code reading to gain more insight in program comprehension	Diese Publikation (Busjahn, Schulte & Busjahn, 2011) beschäftigt sich in ihrer Studie mit den Unterschieden, die beim Lesen von Sourcecode und bei natürlichem Text auftreten.
Bednarik	2012	Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations	Es wird die Effektivität von multiplen Repräsentationsformen während des Debuggings untersucht (Bednarik, 2012).

Autoren	Jahr	Titel	Zusammenfassung
Hejmady & Narayanan	2012	Visual attention patterns during program debugging with an IDE	Die Autoren (Hejmady & Narayanan, 2012) untersuchen die Effektivität und Bedeutung von multiplen Repräsentationsformen während des Debuggings und den Einfluss von Expertise.
Sharafi, Soh, Guéhéneuch & Antoniol	2012	Women and men - different but equal: On the impact of identifier style on source code reading	Eine Forschungsgruppe (Sharafi, Soh, Gueheneuc & Antoniol, 2012) erforscht, wie sich verschiedene Identifier-Stile (camel case vs. Underscore) auf die Erinnerung der Entwicklerinnen und Entwickler an diese auswirken. Ergänzend werden geschlechterspezifische Strategien verglichen.
Sharif, Falcone & Maletic	2012	An eye-tracking study on the role of scan time in finding source code defects	Die Studie (Sharif, Falcone & Maletic, 2012) repliziert teilweise die Studie von Uwano et al. (2006) mit einer größeren Stichprobe und ergänzenden Eye-Tracking-Metriken. Es wird auch in dieser Studie untersucht, wie die Probanden einen Code nach Fehlern durchsuchen.
Binkley, Davis, Lawrie, Maletic, Morell & Sharif	2013	The impact of identifier style on effort and comprehension	Die Autoren (Binkley et al., 2013) beschäftigen sich mit den Auswirkungen von verschiedenen Identifier-Stilen auf das Verständnis von Code. Sie fassen in ihrer Arbeit die Ergebnisse aus zwei verschiedenen Studiensettings zusammen (Erinnerungs- und Multiple-Choice-Fragen).
Hou, Lin, Lin, Chang & Yen	2013	Exploring the gender effect on cognitive processes in program debugging based on eye movement analysis	Diese Studie (Hou, Lin, Lin, Chang & Yen, 2013) fokussiert geschlechtsspezifische kognitive Unterschiede beim Debugging. Die Analyse der Augenbewegungen zeigt, entsprechende Unterschiede auf. So benötigen Frauen bei rekursiven Problemen mehr manuelle Berechnungen, während Männer iterative Probleme logikbasiert angehen.

Autoren	Jahr	Titel	Zusammenfassung
Sharif, Jetty, Apnte & Parra	2013	An empirical study assessing the effect of SeeIT3D on comprehension	Die Autoren (Sharif, Jetty, Aponte & Parra, 2013) untersuchen die Usability eines Visualisierungstools namens SeeIT um code overview und bug-fixing-Aufgaben auszuführen.
Busjahn, Bednarik & Schulte	2014	What influences dwell time during source code reading?	Diese Studie adressiert die Aufmerksamkeitsverteilung auf verschiedene Code-Elemete um Unterschiede zwischen den Lesestrategien von Experten und Novizen zu identifizieren (Busjahn, Bednarik & Schulte, 2014).
Fritz, Begel, Müller, Yigit-Elliott & Züger	2014	Using psychophysiological measures to assess task difficulty in software development	Im Falle dieser Arbeit wird ein Mixed-Methods-Ansatz verwendet, der Eye-Tracking, Elektroenzephalographie (EEG), elektrodermale Aktivitäten (EDA) und NASA TLX-Fragebogen kombiniert, um die Schwierigkeit von Programmieraufgaben zu untersuchen (Fritz, Begel, Müller, Yigit-Elliott & Züger, 2014).
Rodeghero, McMillan, McBurney, Bosch & D'Mello	2014	Improving automated source code summarization via an eye-tracking study of programmers	Es wird eine Eye-Tracking-Studie durchgeführt, deren Ergebnisse zur Entwicklung eines Tool zur Code-Zusammenfassung genutzt werden (Rodeghero, McMillan, McBurney, Bosch & D'Mello, 2014).
Turner, Falcone, Sharif & Lazar	2014	An eye-tracking study assessing the comprehension of C++ and Python source code	Es wird die Verständlichkeit von Quellcodes in C++ und Python miteinander verglichen, indem die Scanpfade von Experten und Novizen gegenüberstellen werden (Turner et al., 2014).

Autoren	Jahr	Titel	Zusammenfassung
Busjahn, Bednarik, Begel, Crosby, Paterson, Schulte, Sharif & Tamm	2015	Eye movements in code reading: Relaxing the linear order	Die Studie (Busjahn et al., 2015) untersucht, wie linear (von link nach rechts und von oben nach unten) Quellcode von Experten und Novizen gelesen wird. Die Ergebnisse belegen, dass beide Gruppen Quellcode weniger linear als natürlichen Text lesen. Weiterhin zeigt sich, dass das nicht-lineare Lesen mit ansteigender Erfahrung zunimmt.
Jbara & Feitelson	2015	How programmers read regular code: A controlled experiment using eye-tracking	Es wird eine Eye-Tracking-Studie durchgeführt welche klären soll, wie wiederkehrende Muster im Code dessen Verständlichkeit beeinflussen. Die Analyse zeigt, dass mehr Zeit und Aufwand in die initialen Code-Abschnitte investiert werden und sich bei Wiederholungen ein signifikanter Abfall beobachten lässt (Jbara & Feitelson, 2015).
Kevic, Walters, Shaffer, Sharif, Shepherd & Fritz	2015	Tracing software developers' eyes and interactions for change tasks	Die Autoren (Kevic et al., 2015) untersuchen in ihrer Studie, wie sich Entwickler verhalten, wenn an einem Open-Source-Programm Änderungen vorgenommen werden sollen. Es zeigt sich, dass sich die Entwickler meist nur auf kleine Teile von Methoden konzentrieren, welche durch einen Datenfluss verbunden sind.
Rodeghero, Liu, McBurney & McMillan	2015	An eye-tracking study of Java programmers and application to source code summarization	Im Rahmen einer Eye-Tracking-Studie (Rodeghero, Liu, McBurney & McMillan, 2015) wird verglichen, wie sich die Zusammenfassung von Quellcode unterscheidet, wenn diese automatisch von einem Programm oder händisch von einem erfahrenen Programmierer erstellt wird.

Autoren	Jahr	Titel	Zusammenfassung
Beelders & du Plessis	2016	The influence of syntax highlighting on scanning and reading behavior for source code	In dieser Studie (Beelders & du Plessis, 2016) wird der Einfluss von Syntax-Highlighting auf das Lesen von Quellcode untersucht. Es zeigt sich, dass unabhängig von der Darstellung des Codes die Phasen, die Programmierer während eines Reviews durchlaufen, vorhanden sind. Die Anwesenheit oder Abwesenheit von Syntaxhervorhebung hat keinen Einfluss auf Fixationsdauer oder -anzahl, und das Leseverhalten unterscheidet sich zwischen den Scanning- und Lesephasen nicht, was darauf hinweist, dass die kognitive Belastung zwischen diesen nicht wesentlich variiert. Somit wirkt sich das Fehlen von Syntax-Highlighting nicht nachteilig aus.
Lin, Wu, Hou, Lin, Yang & Chang	2016	Tracking students cognitive processes during program debugging an eye-movement approach	Die Autoren (Lin et al., 2016) setzen in ihrer Studie einen Eye-Tracker ein um die kognitiven Prozesse von Studierenden beim Debuggen von Programmen zu analysieren. Die Ergebnisse zeigen, dass leistungsstarke Studenten beim Debuggen die Programme logischer verfolgten, während leistungsschwache Studenten oft Zeile für Zeile vorgingen und die übergeordnete Logik des Programms langsamer erfassen konnten. Letztere benötigten mehr Zeit für manuelle Berechnungen und sprangen oft direkt zu vermuteten Auffälligkeiten, ohne der Logik des Programms zu folgen.
Nivala, Hauser, Mottok & Gruber	2016	Developing visual expertise in software engineering: An eye tracking study	Diese Studie (Nivala et al., 2016) beschäftigt sich mit erfahrungsbedingten Unterschieden bei Debugging- und Verständnisaufgaben. Die Ergebnisse zeigen, dass Novizen mehr Zeit mit dem Lesen des Codes als mit dem Verfassen der Antwort verbringen, während die fortgeschrittene Programmierer früher mit dem Verfassen beginnen und mehr Zeit für diese aufwenden. Weiterhin haben die Fortgeschrittenen kürzere Fixationen und Sakkaden. Die Ergebnisse deuten darauf hin, dass fortgeschrittene Probanden schneller den Inhalt des Codes erfassen und mehr Details erkennen können.

Autoren	Jahr	Titel	Zusammenfassung
Peng, Li, Shong, Hu & Feng	2016	An eye tracking research on debugging strategies towards different types of bugs	Die Studie (Peng, Li, Song, Hu & Feng, 2016) widmet sich der Erforschung von Debugging-Strategien. So zeigt sich, dass es bei Datenflussfehlern vorteilhaft ist, auf Änderungen von Variablen zu achten. Bei Kontrollflussfehlern hingegen ist es wichtiger, den Code zu beobachten und die logische Struktur zu verstehen. Diese Erkenntnisse in Kombination mit Fehlermeldungen vom Compiler können Programmierern dabei helfen, Defekte effizienter zu finden.
Barik, Smith, Lubick, Holmes, Feng, Murphy-Hill & Parnin	2017	Do Developers Read Compiler Error Messages?	In ihrer Publikation untersuchen die Autoren (Barik et al., 2017) den Umgang von Entwicklern mit Fehlermeldungen in einer IDE. Es zeigte sich, dass diese Meldungen in ihrer Komplexität mit Quellcode vergleichbar sind und von den Versuchspersonen gelesen werden. Ebenso können Schwierigkeiten beim Lesen der Fehlermeldungen als Prädikator für die Leistung der Versuchspersonen genutzt werden. Weiterhin kann beobachtet werden, dass die Entwickler einen großen Teil ihrer Bearbeitungszeit (ca. 13-25%) mit dem Lesen von Fehlermeldungen verbringen.
Kevic	2017	Using eye gaze data to recognize task-relevant source code better and more fine-grained	Diese Arbeit (Kevic, 2017) beschäftigt sich mit der Verbesserung von Recommender-Systemen durch die Einbeziehung von Eye-Tracking-Daten. Die Studie zeigt, zwei zentrale Resultate: Auf der einen Seite wird deutlich, dass die Einbeziehung der Augenbewegung einen signifikanten Beitrag zur Verbesserung der Systeme leistet, als dies beispielsweise nur durch die Analyse der Mausklicks erfolgen könnte. Auf der anderen Seite liefert sie Erkenntnisse zur Vorgehensweise der Programmierer. Die Ergebnisse zeigen, dass die Probanden lediglich 11% der Zeilen innerhalb einer Methode als relevant erachten. Insgesamt betont die Studie die Erkenntisse, dass die Augenbewegungsdaten mehr Auskunft über die kognitiven Prozesse liefern, als dies Mausbewegungen machen.

Autoren	Jahr	Titel	Zusammenfassung
Melo, Narcizo, Hansen, Brabrand & Wasowski	2017	Variability through the eyes of the programmer	Im Falle dieser Studie (Melo, Narcizo, Hansen, Brabrand & Wasowski, 2017) wird untersucht, wie Entwickler Programme mit und ohne Variabilität debuggen. Die Ergebnisse zeigen, dass die Debugging-Zeit für Code-Abschnitte mit Variabilität zunimmt. Weiterhin scheint die Debugging-Zeit auch für Code-Abschnitte ohne Variabilität in der Nähe von Abschnitten mit Variabilität zuzunehmen. Die Anwesenheit von Variabilität korreliert mit einer Zunahme der Blickübergänge zwischen Definitionen und Verwendungen von Feldern und Methoden. Variabilität verlängert auch den ersten Scan des gesamten Programms.
Peachock, Iovino & Sharif	2017	Investigating Eye Movements in Natural Language and C++ Source Code: A Replication Experiment	Die Autoren (Peachock, Iovino & Sharif, 2017) beschäftigen sich mit erfahrungsbedingten Unterschieden beim Lesen von natürlichen Texten und Quellcode. Dazu replizieren sie die Arbeit von Busjahn et al. (2015), nutzen jedoch C++ anstatt von Java als Programmiersprache. Die Ergebnisse zeigen, dass sowohl Anfänger, als auch erfahrene Programmierer Quellcode weniger linear lesen als natürliche Texte. Es konnten weder für das Lesen von natürlichen Texten, noch für das Lesen von Quellcode signifikante Unterschiede zwischen den beiden Erfahrungsstufen festgestellt werden.
Begel & Vrzakova	2018	Eye Movements in Code Review	Begel & Vrzakova (2018) gehen der Frage nach, welche Augenbewegungen von Softwareentwicklerinnen und -entwicklern während eines Code Reviews gezeigt werden. Im Rahmen ihrer Arbeit identifizieren sie verschiedene Muster und halten fest, dass sich Code Reviews vorrangig aus *skimming* und *careful reading* zusammensetzen, welches sich immer wieder abwechseln.

Autoren	Jahr	Titel	Zusammenfassung
Chandrika, Amudha & Sudarsan	2018	Recognizing Eye Tracking Traits for Source Code Review	Eine Forschungsgruppe (Chandrika, Amudha & Sudarsan, 2018) beschäftigt sich mit dem Debugging von C#-Codes und der textuellen Beschreibung eines Algorithmus. Es konnte beobachtet werden, dass die erfahrenen Probanden eine deutlich bessere Performance bei der Fehlersuche zeigen und insgesamt effizienter mit dem Code arbeiten. Die Autoren kommen zu dem Schluss, dass die Aufmerksamkeit auf fehlerhafte Zeilen und Kommentare, sowie code coverage zentrale Aspekte für ein erfolgreiches Review sind.
Konopka, Talian, Tvarozek & Navrat	2018	Data Flow Metrics in Program Comprehension Tasks	Aus Sicht der Verfasser (Konopka, Talian, Tvarozek & Navrat, 2018) hat die bisherige Forschung zur *program comprehension* zu wenig Beachtung auf die Abdeckung von Programmierkonzepten gelegt und weiterhin mangelt es an entsprechenden Metriken um diese strukturiert erfassen zu können. Um diese Lücke zu schließen, werden in der Arbeit der Autoren verschiedene grundlegende Programmierkonzepte untersucht Konopka et al. (2018) machen den Vorschlag zukünftig *Data Flow Metrics* zu nutzen um das Leseverhalten besser erfassen zu können.
Mangaroska, Sharma, Giannakos, Trætteberg & Dillenbourg	2018	Gaze Insights into Debugging Behavior Using Learner-Centred Analysis	Die Autoren (Mangaroska, Sharma, Giannakos, Trætteberg & Dillenbourg, 2018) untersuchen in ihrer Studie, wie Nutzerdaten aus der IDE Eclipse dazu genutzt werden können um das studentische Lernen zu verbessern. Durch das Plug-In *Excercise View* werden während einer Debugging-Aufgabe die Blickbewegungen der Versuchspersonen erfasst. Eine zentrale Erkenntnis ist, dass Studierende, die die zusätzlichen Informationen nutzen, ihre Leistung verbessern und größeren Erfolg erzielen. Ebenso betonen die Autoren die Bedeutung von Eye-Tracking-Daten für die Analyse und Gestaltung von datengetriebenen Lernumgebungen.

Autoren	Jahr	Titel	Zusammenfassung
Abid, Maletic & Sharif	2019	Using Developer Eye Movements to Externalize the Mental Model Used in Code Summarization Tasks	Die Autoren (Abid, Maletic & Sharif, 2019) untersuchen die Augenbewegungen von Entwicklerinnen und Entwicklern um mentale Modelle zum Verständnis von Quellcode zu erforschen. Die Studie zeigt, dass Experten und Anfänger im Durchschnitt die Methoden genauer lesen (mit dem *bottom-up Modell*) als hin und her springen (mit *top-down*). Allerdings verbringen Novizen im Vergleich zu den Experten Durchschnitt mehr Zeit mit *bottom-up*-Ansätzen.
Obaidellah, Raschke & Blascheck	2019	Classification of Strategies for Solving Programming Problems using AoI Sequence Analysis	Die Studie von Obaidellah et al. (2019) nutzt Eye-Tracking, um die visuelle Aufmerksamkeit von Teilnehmern bei der Lösung von Programmieraufgaben zu analysieren. Die Auswertung der Augenbewegungen deutet darauf hin, dass die Teilnehmer in effektive und ineffektive Problemlöser unterteilt werden können, was auf unterschiedliche kognitive Schemata und Leistungen hinweist.
Peterson, Abid, Bryant, Maletic & Sharif	2019	Factors influencing dwell time during source code reading	Diese Arbeit (Peterson, Abid, Bryant, Maletic & Sharif, 2019) beschäftigt sich mit Einflussfaktoren auf die *dwell time* während eines Code Reviews und nutzt dafür ein erweitertes Design der Studie von Busjahn et al. (2014). Aufgabe der Versuchspersonen ist es, Methoden schriftlich zusammenzufassen. Im Rahmen der Analyse wurden keine Korrelation zwischen Zeilenlänge und der gesamten auf die Zeile gerichteten Betrachtungsdauer festgestellt. Die ersten Fixierungen innerhalb einer Methode beziehen sich meist auf die Signatur einer Methode, eine Variablendeklaration oder eine Zuweisung im Vergleich zu anderen Fixierungen innerhalb einer Methode. Darüber hinaus wurde festgestellt, dass kleinere Methoden tendenziell eine kürzere *dwell time* für die gesamte Methode aufweisen, jedoch eine signifikant längere pro Zeile in der Methode.

Autoren	Jahr	Titel	Zusammenfassung
Saddler	2019	Looks Can Mean Achieving: Understanding Eye Gaze Patterns of Proficiency in Code Comprehension	Diese Studie widmet sich in seiner Studie dem Zusammenhang von Blickrichtung und Verständnis bei Quellcode. Es zeigt sich, dass Programmieranfängerinnen und -anfänger Ausgabeanweisungen und Deklarationen genauso oft besuchen wie den restlichen Code (Saddler, 2019).

Tabelle 3.1: Auflistung aller relevanter Studien mit Kurzzusammenfassung

Die Auflistung aller relevanten Studien in Tabelle 3.1 liefert grundlegende Informationen zum aktuellen Forschungsstand dieser Dissertation und zeigt auf, an welchen Stellen noch Lücken existieren. Obwohl Erfahrung und Expertise bereits bezüglich der Erforschung visueller Expertise bei Code Reviews in Studien aufgenommen worden ist, zeigt sich, dass dies meist unsystematisch geschah und die Einteilungen in Experten und Novizen nicht theoriegeleitet begründet wurde.

Tabelle 3.1 liefert weiterhin auch Erkenntnisse über die verschiedenen Forscherinnen und Forscher, welche in diesem Feld agieren und wie diese vernetzt sind. Auf der einen Seite lassen sich drei Forschungsgruppen bzw. Forscher identifizieren, welche bereits seit längerer Zeit aktiv sind und ihren Schwerpunkt auf Eye-Tracking-Studien im Software Engineering beziehungsweise auf Code Reviews gelegt haben:

- *Roman Bednarik:* Die Gruppe um Roman Bednarik im finnischen Joensuu ist bereits seit 2006 aktiv und legt ihren Fokus bezüglich Eye-Tracking auf verschiedene Themen des Software Engineerings mit einem starken Interesse an Code Reviews. Wie sich während eines Besuchs an der *University of Eastern Finland* im Jahr 2023 zeigte, hat die Gruppe ihr Themengebiet erweitert und befasst sich ergänzend mit der Anwendung von Eye-Tracking in der Medizin und verschiedenen Aspekten des *language learnings*.

- *Theresa Busjahn:* Theres Busjahn und ihre Kolleginnen und Kollegen befassen sich vorrangig mit Augenbewegungen während eines Code Reviews. Die Gruppe ist vorrangig in Deutschland aktiv und über mehrere Bundesländer verteilt. Im Falle vieler Studien wird auf interdisziplinäre Ansätze gesetzt, bei denen verschiedene Theorien und Methoden kombiniert werden (Busjahn et al., 2015).

- *Bonita Sharif:* Die inzwischen an der *University of Nebraska-Lincoln* aktive Professorin Bonita Sharif beschäftigt sich ebenfalls primär mit der Anwendung von Eye-Tracking auf Code Reviews. Sie und ihre Kolleginnen und Kollegen führen auf der einen Seite Studien durch, entwicklen aber auch Tools, welche hinsichtlich entsprechender Experimente genutzt werden können (Sharif et al., 2013). Zu nennen wäre hier *iTrace*, welches auch im Regensburger Eye-Tracking-Classroom genutzt wird.

Die Liste der aktiven Gruppen ließe sich noch erweitern, jedoch decken die drei genannten Forscherinnen und Forscher, sowie ihre Kolleginnen und Kollegen bereits ein breites Spektrum an Arbeiten ab, welche für diese Dissertation von Bedeutung sind. Ebenso stellen sie hinsichtlich Vernetzungsaktivitäten zentrale Akteure dar. Anzumerken ist auch, dass der gesamte Forschungssektor zur Anwendung von Eye-Tracking auf Code Reviews ein relativ harmonisches Bild abgibt. So zeigen sich bei Betrachtung der in Tabelle 3.1 dargelegten Studien viele Kooperationen zwischen den Forscherinnen und Forschern. Dies wird auch durch gemeinsam organisierte Konferenzen (allen voran zu nennen wäre hier der inzwischen jährlich stattfindende Workshop *Eye Movements in Programming*) verdeutlicht und zeigt auch auf, dass die thematische Relevanz in den letzten Jahren eher zugenommen hat.

3.2.4 Zeitliche Abfolge der relevanten Publikationen

Ein Indikator, welcher ebenfalls in den systematischen Literaturvergleich aufgenommen wird, sind die thematisch relevanten Publikationen pro Jahr. Auf der einen Seite bietet diese Metrik einen Überblick über die zeitliche Abfolge der relevanten Studien, auf der anderen Seite dient sie auch als Indikator für das Forschungsinteresse für den Einsatz von Eye-Tracking zur Analyse von Code Review. Diesbezüglich wurde der gesamte Zeitraum von 1990 bis 2019 betrachtet. Eine grafische Darstellung findet sich in Abbildung 3.2, welche nachfolgend näher beschrieben werden soll:

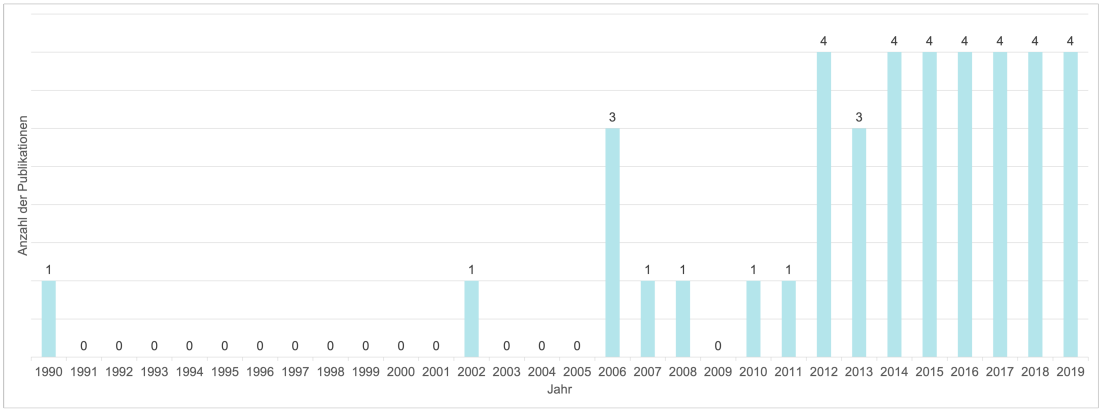

Abbildung 3.2: Publikationen pro Jahr

Die erste relevante Arbeit findet sich in der 1990 von Crosby und Stelovsky publizierten Studie *How do we read algorithms? A case study*. Sie bildet de facto den Startpunkt dieses Forschungsbereichs. Bei der Betrachtung der Zeitachse ist es jedoch etwas verwunderlich, dass die nächste Publikation erst 2002 erschienen ist. Von einem gesteigerten Forschungsinteresse in diesem Bereich lässt sich erst ab 2012 sprechen. Ab diesem Jahr erschienen (bis auf eine Ausnahme im Jahr 2013) meist 4 thematisch relevante Studien für den Kontext dieser Dissertation. Somit bildet sich auch ein Trend hin zu einem verstärkten Forschungsinteresse für die Analyse von Augenbewegungen während eines Code Reviews.

Auch diese Entwicklung lässt sich mit der Verfügbarkeit und technischen Entwicklung von Eye-Trackern (Obaidellah et al., 2018; Sharafi et al., 2015) erklären. Diese wurden seit den 1990er-Jahren nicht nur in deutlich größeren Stückzahlen gefertigt, sondern auch immer präziser und somit interessanter für die Untersuchung von Quellcode (Lemahieu & Wyns, 2010; Obaidellah et al., 2018; SensoMotoric Instruments, 2017; Sharafi et al., 2015; Tobii Pro, 2020, 2021).

3.2.5 Verwendete Stimuli

Die Sichtung der ausgewählten Studien zeigt, dass sich hinsichtlich der verwendeten Stimuli bzw. Programmiersprache Java als klarer Favorit präsentiert. Diese Sprache wird in mindestens 16 der relevanten Publikationen genutzt. Ebenso erfreuen sich verschiedene Sprachen aus der C-Familie (8 Studien) einer gewissen Beliebtheit und werden für Experimentalzwecke genutzt. Schwieriger gestaltet es sich bei den

restlichen Publikationen. Diese geben zum Teil keine Informationen darüber, welche Sprache genau verwendet wurde oder gehen nur vage auf die verwendeten Beispiele ein. Häufig ist dies auch der Fall, wenn mit vollständigen Programmen oder open-source-Projekten gearbeitet worden ist (Mangaroska et al., 2018; Peterson et al., 2019; Rodeghero et al., 2014). Ebenfalls als schwierig stellt sich die Kombination dar, bei der Code zum Teil mit grafischen Elementen (z.B. Diagrammen), natürlicher Sprache oder einer anderen Programmiersprache verglichen worden ist (Binkley et al., 2013; Sharafi et al., 2012; Turner et al., 2014). Hier ist es häufig nicht eindeutig möglich zu untersuchen, worauf der Fokus lag. Eine Übersicht über die verwendeten Programmiersprachen bietet die Abbildung 3.3.

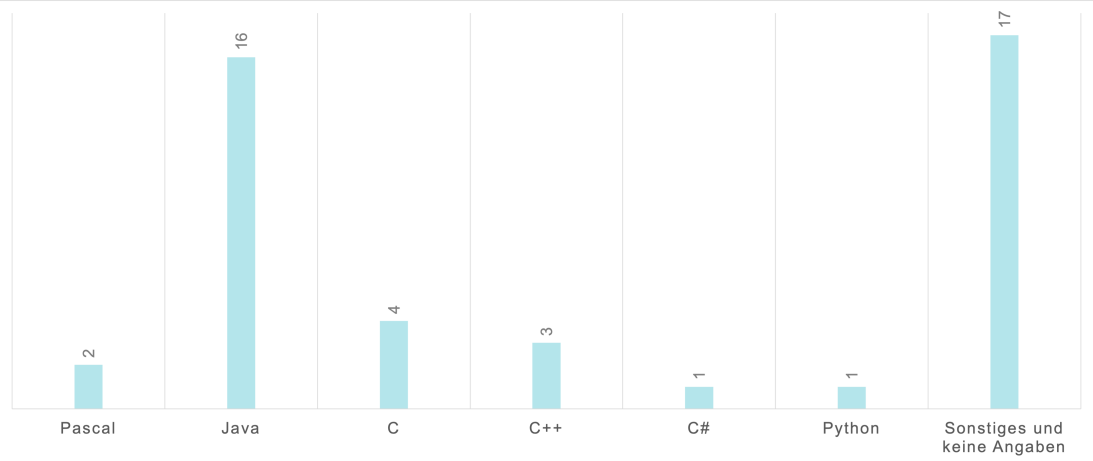

Abbildung 3.3: Verwendete Programmiersprachen in den ausgewählten Studien

3.2.6 Untersuchte Stichproben

Eine Betrachtung der Stichprobengrößen in den ausgewählten Studien zeigt dass sich diese meist in einem Rahmen von ca. 15-35 Versuchspersonen bewegt, wodurch die Stichproben beispielsweise im Vergleich zu Fragebogenstudien eher gering ausfallen. Im Durchschnitt ergibt sich für die analysierten Studien ein Mittelwert von rund 27.375 Versuchspersonen. Diese verhältnismäßig kleine Größe wurde bereits von Sharafi et al. 2015) thematisiert und mit der eher aufwändigen Testung, der Verfügbarkeit von Eye-Trackern und den Teilnahmevoraussetzungen für die jeweiligen Studien begründet. Gerade in Bezug auf den letztgenannten Punkt gilt es zu beachten, dass Versuchspersonen im Großteil der Fälle über spezielle Fähigkeiten und Wissen verfügen müssen um an den Studien teilnehmen zu können.

Ausreißer ergeben sich sowohl in Bezug auf besonders große, als auch in Bezug auf besonders kleine Stichproben. Die größte Stichprobe findet sich in der Arbeit von Binkley et al. (2013) mit insgesamt 169 Probanden. Im Falle dieser Arbeit sollte jedoch beachtet werden, dass Binkley et al. (2013) weniger von Personen, sondern vielmehr von "Fällen" sprechen. So durchliefen die gleichen Versuchspersonen verschiedene Experimente mehrfach und wurden aufaddiert. Strenger bewertet stellt sich die Arbeit von Sharif et al. (2013) mit dem Titel *An empirical study assessing the effect of SeeIT*

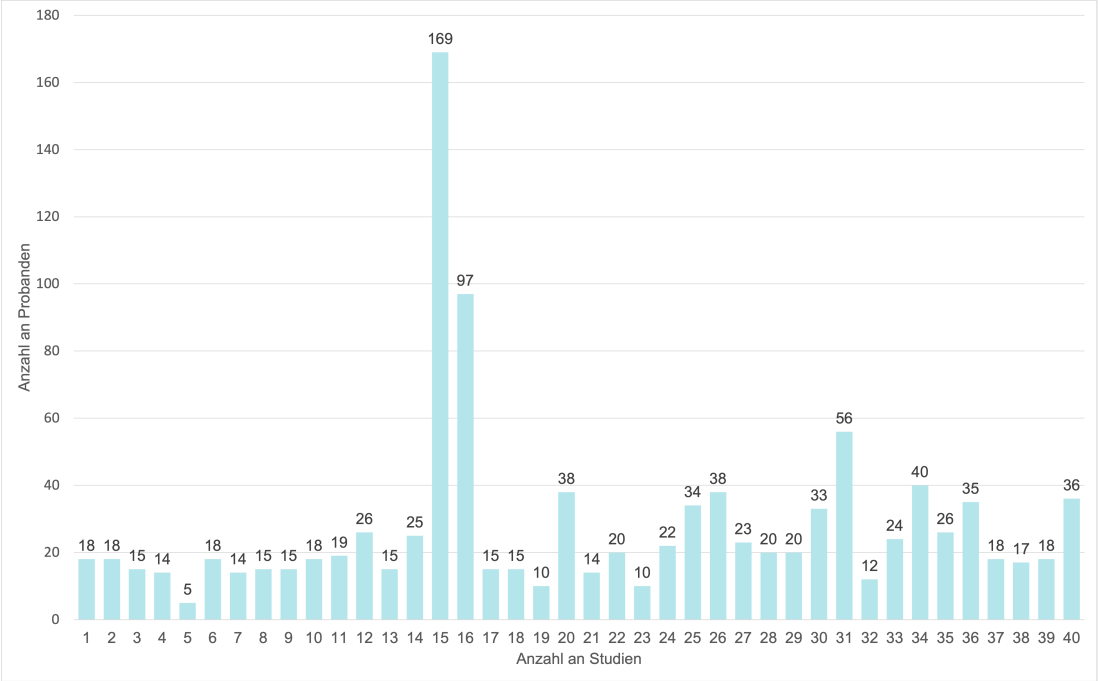

Abbildung 3.4: Stichprobengrößen der analysierten Studien

3D on comprehension dar. Für diese Studie konnte auf insgesamt 97 Studierende von drei Universitäten zurückgegriffen werden, wodurch sich somit die für diesen Literaturvergleich größte Stichprobe an natürlichen Personen ergibt. Ebenfalls erwähnt sei auch an dieser Stelle die Arbeit von Barik et al. (2017), welche insgesamt 56 Versuchspersonen rekrutieren konnten. Die kleinste Stichprobe findet sich bei Uwano et al. (2006), welche in ihrer Studie *"Analyzing individual performance of source code review using reviewer's eye movement"* lediglich fünf Versuchspersonen rekrutieren konnten.

Eine grafische Darstellung der Stichprobengrößen findet sich in der Abbildung 3.4. Diese verdeutlicht, wie sich die durchschnittliche Stichprobe zusammensetzt und illustriert weiterhin die angesprochenen Ausreißer.

Betrachtet man die Stichprobengrößen in Bezug auf die zeitliche Abhängigkeit, so zeigt sich, dass seit ca. 2014 zunehmen. Dies mag unterschiedliche Gründe haben, ist aber sicherlich mit der besseren Verfügbarkeit von professionellen Eye-Trackern, sowie dem gesteigerten Forschungsinteresse an diesem Themenbereich zu verbinden (siehe diesbezüglich auch die gesteigerte Anzahl an Publikationen bei 3.2.3).

3.2.7 Quellen der ausgewählten Studien

Im Rahmen des Literaturvergleichs wird auch dargestellt, aus welchen Quellen die analysierten Studien stammen. Tabelle 3.2 bietet diesbezüglich eine detaillierte Übersicht und soll nachfolgend näher erklärt werden.

Tabelle 3.2: Auflistung der relevanten Konferenz- und Zeitschriftenbeiträge

Quelle	Anzahl	Publikationen

Tabelle 3.2: Auflistung der relevanten Konferenz- und Zeitschriftenbeiträge

Quelle	Anzahl	Publikationen
Behavior research methods	1	Bednarik & Tukiainen (2007)
Computer	1	Crosby & Stelovsky (1990)
Empirical Software Engineering	1	Binkley, Davis, Lawrie, Maletic, Morell & Sharif (2013)
IEEE Transactions on Education	1	Lin, Wu, Hou, Lin, Yang & Chang (2016)
IEEE Transactions on Software Engineering	1	Rodeghero, Liu, McBurney & McMillan (2015)
International Journal of Human Computer Studies	1	Bednarik (2012)
Proceedings of the ACM International Conference on Learning Analytics and Knowledge (LAK)	1	Mangaroska, Sharma, Giannakos, Trætteberg & Dillenbourg (2018)
Proceedings of the ACM Symposium on Eye tracking research & applications (ETRA)	11	Bednarik & Tukiainen (2006); Uwano, Nakamura, Monden & Matsumoto (2006); Bednarik & Tukiainen (2008); Hejmady & Narayanan (2012); Sharif, Falcone & Maletic (2012); Busjahn, Bednarik & Schulte (2014); Turner, Falcone, Sharif & Lazar (2014); Peterson, Abid, Bryant, Maletic & Sharif (2019); Saddler (2019); Abid, Maletic & Sharif(2019); Obaidellah, Raschke & Blascheck (2019)
Proceedings of the ACM Workshop on Eye Movements in Programming (EMIP)	2	Begel & Vrzakova (2018); Konopka, Talian, Tvarozek & Navrat (2018)
Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT)	1	Beelders & du Plessis (2016)
Proceedings of the Augmented Cognition (AC)	1	Peachock, Iovino & Sharif (2017)

Tabelle 3.2: Auflistung der relevanten Konferenz- und Zeitschriftenbeiträge

Quelle	Anzahl	Publikationen
Proceedings of the Hawaii International Conference on System Sciences (HICSS)	1	Aschwanden & Crosby (2006)
Proceedings of the IEEE Annual Computer Software and Applications Conference (COMPSAC)	1	Peng, Li, Shong, Hu & Feng (2016)
Proceedings of the IEEE Global Engineering Education Conference (EDUCON)	1	Nivala, Hauser, Mottok & Gruber (2016)
Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)	1	Chandrika, Amudha & Sudarsan (2018)
Proceedings of the IEEE International Conference on Program Comprehension (ICPC)	5	Sharif & Maletic (2010); Sharafi, Soh, Guéhéneuch & Antoniol (2012); Busjahn, Bednarik, Begel, Crosby, Paterson, Schulte, Sharif & Tamm (2015); Jbara & Feitelson (2015); Melo, Narcizo, Hansen, Brabrand & Wasowski (2017)
Proceedings of the IEEE Working Conference on Software Visualization (VISSOFT)	1	Sharif, Jetty, Apnte & Parra (2013)
Proceedings of the International Conference on Computer Supported Education	1	Hou, Lin, Lin, Chang & Yen (2013)
Proceedings of the ACM/IEEE International Conference on Software Engineering (ICSE)	4	Fritz, Begel, Müller, Yigit-Elliott & Züger (2014); Rodeghero, McMillan, McBurney, Bosch & D'Mello (2014); Barik, Smith, Lubick, Holmes, Feng, Murphy-Hill & Parnin (2017); Kevic (2017)
Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)	1	Kevic, Walters, Shaffer, Sharif, Shepherd & Fritz (2015)
Proceedings of the Koli Calling International Conference on Computing Education Research	1	Busjahn, Schulte & Busjahn (2011)
Proceedings of the Workshop of the Psychology of Programming Interest Group	1	Crosby, Scholz & Wiedenbeck (2002)

Tabelle 3.2: Auflistung der relevanten Konferenz- und Zeitschriftenbeiträge

Quelle	Anzahl	Publikationen

Es zeigt sich, dass ein Großteil der aufgelisteten Studien (34 von 40) bei Konferenzen vorgestellt und in den jeweiligen Konferenzbänden erfasst wurden. Besonders stark vertreten sind diesbezüglich das *ACM Symposium on Eye Tracking Research & Applications (ETRA)* mit elf Publikationen, die *IEEE International Conference on Program Comprehension (ICPC* mit fünf Beiträgen, sowie die (meist) gemeinsam von ACM und IEEE organisierte *International Conference on Software Engineering (ICSEE)* mit fünf Publikationen. Zwar noch nicht so stark im relevanten Forschungsstand vertreten, jedoch stark auf das Themenfeld der Dissertation ausgerichtet, ist der ebenfalls von ACM (und meist begleitend zur *ETRA*) organisierte *Workshop on Eye Movements in Programming (EMIP)*, welcher bereits als Basis für Publikationen des hier dargestellten Literaturvergleichs dient.

Bezüglich Zeitschriften zeigt sich ein sehr verteiltes Bild. Diese machen insgesamt sechs der 40 analysierten Studien aus, jedoch ergibt sich hinsichtlich beliebter Journale für die Publikation von Eye-Tracking-Studien kein Favorit. Alle sechs Studien wurden bei unterschiedlichen Zeitschriften veröffentlicht. Bei diesen handelt es sich um *Behavior research methods*, *Computer*, *Empirical Software Engineering*, *IEEE Transactions on Education*, *IEEE Transactions on Software Engineering* und das *International Journal of Human Computer Studies*.

3.2.8 Eingesetzte Eye-Tracker

Die in den Studien des Literaturvergleichs eingesetzten Eye-Tracker stellen ebenfalls einen Bestandteil dieser Betrachtung dar. Übergreifend über alle Studien zeigt sich eine deutliche Dominanz der Geräte des schwedischen Herstellers Tobii. Insgesamt wurden 23 Eye-Tracker von Tobii im Kontext der analysierten Studien verwendet. Der am häufigsten verwendete Eye-Tracker ist in diesem Fall der Tobii 1750, welcher in sieben Studien zum Einsatz kam. Ebenfalls eher häufiger eingesetzte Eye-Tracker stammen von SensoMotoric Instruments. Die Geräte des deutschen Herstellers, welcher inzwischen von Apple aufgekauft worden ist, wurden insgesamt in vier Studien eingesetzt, wobei hier der SMI 250RED in zwei Studien genutzt wurde um Augenbewegungen zu erfassen.Nennenswert sind auch noch die Geräte von SR Research, deren Eyelink 1000 ebenfalls für zwei Studien genutzt wurde. Selbiges gilt für den GP3 vom Hersteller GazePoint. Eine detaillierte Auflistung der verwendeten Eye-Tracker findet sich in Abbildung 3.5.

Eine Betrachtung der verwendeten Eye-Tracker ist insofern von Interesse, da die Erfassung bestimmter Eye-Tracking-Metriken (z.B. verschiedene Arten von Saccaden) Anforderungen an die Hardware stellt. So werden Saccaden erst ab ca. 120Hz und Micro-Saccaden erst ab ca. 500Hz sichtbar (Duchowski, 2017; Holmqvist et al., 2011;

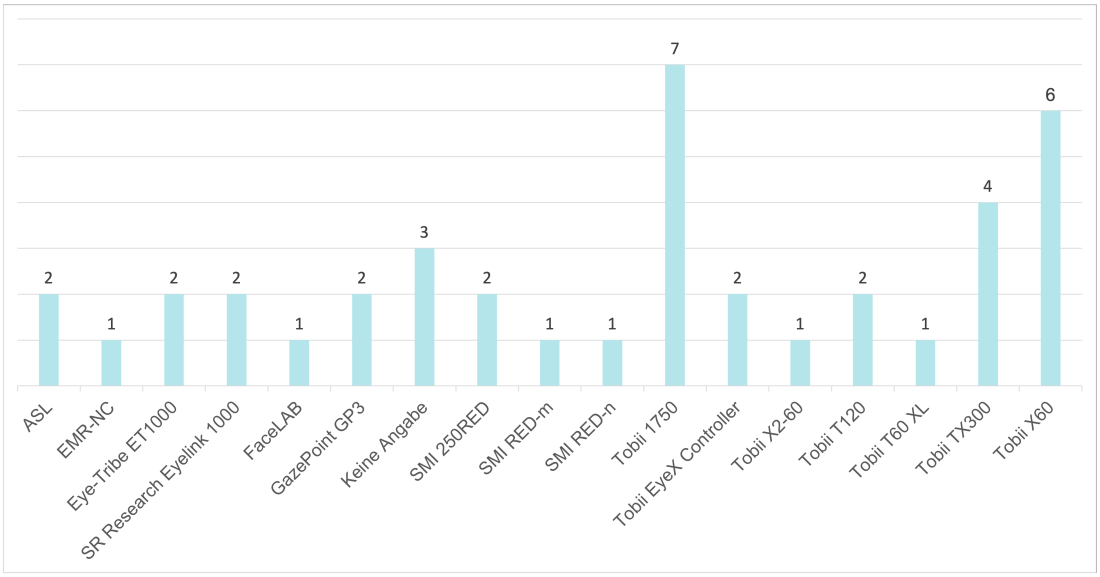

Abbildung 3.5: Eingesetzte Eye-Tracker

Holmqvist & Andersson, 2017). Ebenso sollte bedacht werden, dass die Erfassung von Augenbewegungen während eines Code Reviews eine relativ hohe Genauigkeit des Eye-Trackers voraussetzt. So bewegen sich beispielsweise der Tobii 1750 und der Tobii X60 mit einer Genauigkeit von 0.5° in einem Bereich, der aus heutiger Sicht eher als ungenau bezeichnet werden kann (Tobii Technology, 2010). Die aus technischer Sicht besser geeigneten Geräte, wie beispielsweise der Eye-Link II von SR Research oder der Tobii TX300 kamen aus Sicht dieses Literaturvergleiches erst verhältnismäßig spät auf den den Markt, und standen den Forscherinnen und Forschern nur bedingt zur Verfügung (Obaidellah et al., 2018; Sharafi et al., 2015). Trotz der zuvor genannten Kritik an der *sampling rate* und der *accuracy* muss natürlich auch erwähnt werden, dass sich die Eye-Tracking-Hardware im letzten Jahrzehnt deutlich verbessert hat. Die angesprochenen Geräte entsprachen zum Zeitpunkt der Durchführung der jeweiligen Studie den technologischen Standards. So lässt sich mit Sicherheit auch über die in dieser Dissertation verwendeten Eye-Tracker (SMI 250RED mobile und Tobii Spectrum) in wenigen Jahren Kritik üben, was in der Folge wieder zeigt, dass sich sowohl die Forschung, als auch die Hardware in diesem Bereich weiterentwickeln (SensoMotoric Instruments, 2017; Tobii Pro, 2020, 2021). Zieht man ein Zwischenfazit zu den verwendeten Eye-Tracker zeigt sich, dass sich die verwendeten Geräte im Laufe der letzen Jahre immer mehr an die Anforderungen in Bezug auf die Untersuchung von Augenbewegungen während eines Code Reviews angepasst haben (Sharafi et al., 2020). Es werden zunehmend genauere und schnellere Geräte eingesetzt, welche es Forscherinnen und Forschern ermöglichen neue Eye-Tracking-Metriken zu untersuchen. Ebenso zeigt sich, dass die Geräte inzwischen verfügbarer werden und die Serienfertigung (wie beispielsweise bei Tobii) einen Beitrag zur Standardisierung und standortübergreifenden Durchführung von Experimenten ermöglichen.

3.3 Fazit zum aktuellen Stand der Forschung

Zusammenfassend lässt sich übergreifend über die beiden Metastudien von Sharafi et al. (2015) und Obaidellah et al. (2018), sowie über den eigenen systematischen Literaturvergleich sagen, dass Software Engineering, insbesondere die Untersuchung von Code Reviews von dieser Technologie profitieren. Durch die Verwendung dieser Technologie haben die Forscher die Möglichkeit ein tieferes Verständnis für die kognitiven Prozesse zu erlangen, die in dieser Domäne in die Bearbeitung der verschiedenen Aufgaben involviert sind. Auch die kontinuierlich steigende Anzahl an Publikationen (siehe Absatz 3.2.4 und Abbildung 3.2) darauf schließen, dass Eye-Tracking inzwischen als eine feste Methode in dieser Domäne angekommen ist und sich ein gesteigertes Forschungsinteresse bemerkbar macht.

Übergreifend über alle Studien, welche in diesem Kapitel analysiert werden zeigt sich jedoch ein zentrales Probleme des Software Engineerings in Bezug auf Eye-Tracking: Die mangelnde Standardisierung. Es macht sich leider bemerkbar, dass es zum jetzigen Zeitpunkt keine einheitlichen Begriffe, Metriken und Methoden gibt. Sharafi et al. (2015) nehmen diesbezüglich in ihrer Metastudie eine deutliche Position ein und fordern, dass die gängigen Standards des Eye-Trackings aus anderen Domänen (Duchowski, 2017; Holmqvist et al., 2011; Holmqvist & Andersson, 2017; Sheridan & Reingold, 2017) für den Bereich des Software Enigeerings adaptiert werden. Weiterhin würde es ihrer Meinung nach auch Sinn machen, wenn standardisierte Guidelines entwickelt und angewandt werden. Erste Versuche wurden diesbezüglich bereits von Bednarik und Tukiainen (2006) unternommen, lassen sich jedoch nur schwer oder gar nicht auf andere Studien übertragen. Einen größeren Beitrag hat Sharafi 2020 diesbezüglich mit Ihren Kolleginnen und Kollegen selbst geleistet, indem sie *A practical guide on conducting eye tracking studies in software engineering* veröffentlicht hat. Dieses Paper dient als ein Leitfaden für das Design von Eye-Tracking-Studien im Software Engineering und bietet eine kompakte Übersicht über die Untergliederung der relevanten Metriken (Sharafi et al., 2020, S.3173-3179). Insgesamt bleibt die Kritik an der mangelnden Standardisierung jedoch noch immer berechtigt. Auch ein Blick in die jeweilige Experimentalsoftware von Tobii, SMI und SR Research zeigt kein einheitliches Bild (SensoMotoric Instruments, 2017; SR Research, 2022; Tobii Pro, 2021). Dies bestätigt, dass es sich bei der mangelnden Standardisierung eher um ein allgemeines Problem des Eye-Trackings handelt. Dennoch darf abgewartet werden, wie sich die Domäne des Software Engineerings diesbezüglich weiterentwickelt und dieser Problematik zukünftig begegnet.

Ein weiteres Problem zeigt sich, bei der Betrachtung der durchgeführten Experimente. Zum Teil sind diese nicht oder nur mit großem Aufwand reproduzierbar. In vielen Fällen werden die verwendeten Artefakte von den Forschern selbst erstellt und der Forschungsgemeinschaft nicht oder bestenfalls nur teilweise zur Verfügung gestellt. Dieses Problem wird jedoch von der Forschungsgemeinde in diesem Bereich verstärkt adressiert: Zum Teil lassen sich inzwischen auch replizierte Studien finden. So wurde beispielsweise die Arbeit von Uwano et al. (2006) teilweise von Sharif et al.

(2012) repliziert. Besonders erwähnenswert ist an dieser Stelle die Arbeit von Roman Bednarik und dessen Kolleginnen und Kollegen (2020), welche im Rahmen des *EMIP* einen Datensatz mit 216 Versuchspersonen zum Thema *Code Comprehension* zur Verfügung stellen. Dieser Datensatz ist frei verfügbar und Forschende in diesem Bereich können diese für ihre Zwecke nutzen.

Kapitel 4

Forschungsfragen und Hypothesenbildung

Basierend auf dem Theoretischen Hintergrund in Kapitel 2 und dem in Kapitel 3 dargelegten Stand der Forschung zu visueller Expertise bzw. zum Einsatz von Eye-Tracking bei Code Reviews, sollen nachfolgend die Ziele definiert werden, welche diese Dissertation verfolgt (siehe Absatz 4.1). Basierend auf den genannten Zielen sollen im Nachgang Forschungsfragen (siehe die Absätze 4.2.1, 4.3.1 und 4.4) formuliert und entsprechende Hypothesen (siehe die Absätze 4.2.2 und 4.3.2) erstellt werden.

4.1 Ziele der Dissertation

Basierend auf den theoretischen Vorarbeiten zu dieser Dissertation lassen sich insgesamt drei zentrale Ziele definieren. Diese lauten folgendermaßen:

1. Identifikation visueller Expertise bei der prozeduralen Programmiersprache C
2. Identifikation visueller Expertise bei der objektorientierten Programmiersprache C++
3. Aufbau eines Modells zur Interpretation visueller Expertise bei Code Reviews

Die drei genannten Ziele sind ineinander verschränkt. So soll visuelle Expertise bei Code Reviews sowohl jeweils durch eine eigene Studie für die prozedurale Programmiersprache C, als auch für die objektorientierte Programmiersprache C++ erfasst und charakterisiert werden. Die aus diesen Studien gewonnenen Erkenntnisse werden in der Folge miteinander verglichen und mit den bereits vorliegenden Erkenntnissen aus dem theoretischen Hintergrund (siehe Kapitel 2) und dem aktuellen Stand der Forschung (siehe Kapitel 3) abgeglichen. Die daraus gewonnenen Erkenntnisse adressieren Ziel 3, welches die Bildung eines Modells der visuellen Expertise bei Code Reviews adressiert.

Hinsichtlich der Ziele 1 und 2 gibt es aufgrund der ähnlichen Studiendesigns mehrere Aspekte zu beachten: Sowohl die im Theorieteil dargelegten Beschreibungen

von visueller Expertise (siehe Absatz 2.3), als auch der aktuelle Stand der Forschung zum Thema dieser Dissertation legen nahe, dass sich erfahrungsbedingte Unterschiede bei der Ausführung eines Code Reviews zwischen Novizinnen und Novizen und zwischen Expertinnen und Experten ergeben werden. Ebenso legen die Vorarbeiten nahe, dass sich diese in der allgemeinen Leistung, aber auch phasenbasiert bemerkbar machen können. Unter Berücksichtigung dieser Gegebenheiten weisen die nachfolgend in den Absätzen 4.2 und 4.3 dargestellten Forschungsfragen und Hypothesen zu beiden Studien entsprechende Ähnlichkeiten auf.

4.2 Formulierung von Forschungsfragen und Hypothesen zur Identifikation von Visueller Expertise bei der prozeduralen Programmiersprache C

Die im Kapitel 5 dargestellte Studie adressiert die Erforschung von visueller Expertise bei der Programmiersprache C. Sie greift in ihrem Design auf einen kontrastiven Vergleich von Novizen und Experten zurück. In den nachfolgenden Absätzen werden die im Kontext dieser Studie relevanten Forschungsfragen (siehe Absatz 4.2.1) und die von diesen abgeleiteten Hypothesen (siehe Absatz 4.2.2) dargestellt.

4.2.1 Forschungsfragen zur Erforschung visueller Expertise bei der prozeduralen Programmiersprache C

Die Studie zur Erforschung von visueller Expertise bei der prozeduralen Programmiersprache C verfolgt zwei zentrale Forschungsfragen:

- Forschungsfrage 1: *Wie unterscheiden sich Novizinnen und Novizen von Expertinnen und Experten hinsichtlich ihrer Augenbewegungen während eines Code Reviews in der Programmiersprache C im Allgemeinen?*
- Forschungsfrage 2: *Welche phasenbasierten Unterschiede machen sich in den Augenbewegungen von Novizinnen und Novizen im Vergleich zu Expertinnen und Experten bei der Durchführung eines Code Reviews in der Programmiersprache C deutlich?*

Forschungsfrage 1 zielt auf die Erfassung von allgemeinen Unterschieden hinsichtlich der Augenbewegungen während eines Code Reviews eines C-Quellcodes ab. In diesem Fall werden vorwiegend aufsummierte Eye-Tracking-Metriken oder deren Durchschnittswerte als abhängige Variable genutzt. Ergänzt werden diese Metriken um weitere Kennzahlen welche die Leistung der Versuchspersonen (beispielsweise die Anzahl der gefundenen Fehler oder die Bearbeitungszeit) darstellen.

Die Beantwortung von Forschungsfrage 2 setzt eine Datenfilterung voraus, welche die Augenbewegungen der Versuchspersonen in verschiedene Phasen unterteilt. Der Vergleich dieser Phasen bedient sich nahezu der gleichen Metriken, welche bereits zur Beantwortung von Forschungsfrage 1 genutzt werden, legt dabei jedoch den

Fokus auf Unterschiede in den Augenbewegungen im Verlauf des Code Reviews. Diese Vorgehensweise ermöglicht Rückschlüsse auf mögliche Strategiewechsel oder -anpassungen während der Bearbeitung der Codebeispiele.

4.2.2 Hypothesen zur Erforschung visueller Expertise bei der prozeduralen Programmiersprache C

Die im vorherigen Absatz genannten Forschungsfragen sollen in einem nächsten Schritt in überprüfbare Hypothesen herunter gebrochen werden. Diese Hypothesen greifen jeweils Metriken des Eye-Trackings auf, welche auf Basis des aktuellen Forschungsstandes als relevant erachtet werden. Besondere Bedeutung kommt dabei den von Sheridan und Reingold (2017,S.5) zusammengefassten Metriken der verschiedenen *holistic models of image perception* zu (siehe Absatz 2.1, welche feingranularer und um Kennzahlen aus der Expertiseforschung (z.B. Performance) ergänzt worden sind.

Für Forschungsfrage 1 ergeben sich insgesamt 13 Hypothesenpaare, die überprüft werden sollen. Diese werden nachfolgend in Tabelle 4.1 aufgelistet:

Tabelle 4.1: Auflistung der allgemeinen Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C

Abhängige Variable	H0	H1
Performance	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht zwischen Novizen und Experten	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten
Total viewing time	Die total viewing time während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total viewing time während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Total number of fixations	Die total number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Average number of fixations	Die average number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Total fixation duration	Die total fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Average fixation duration	Die average fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten

Tabelle 4.1: Auflistung der allgemeinen Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C

Abhängige Variable	H0	H1
Fixation rate	Die fixation rate während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die fixation rate während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Total number of saccades	Die total number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Average number of saccades	Die average number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Total number of visits on erroneous lines	Die total number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Average number of visits on erroneous lines	Die average number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Total dwell time on erroneous lines	Die total dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten
Average dwell time on erroneous lines	Die average dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich signifikant zwischen Novizen und Experten

In Bezug auf Forschungsfrage 2 ergibt sich zusätzlich zur Betrachtung der erfahrungsbedingten Unterschiede zwischen Experten und Novizen noch eine phasenbasierte Betrachtung der gezeigten Augenbewegungen während der Code Reviews. Die stärkere Einbeziehung zeitlicher Veränderungen in den Review-Prozess hat auch daher auch Auswirkungen auf die Formulierung der zu überprüfenden Hypothesenpaare. Im Falle von Forschungsfrage 2 ergeben sich 12 Hypothesenpaare, die nachfolgend beschrieben werden:

Tabelle 4.2: Auflistung der phasenbasierten Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C

Abhängige Variable	H0	H1
Total viewing time	Die total viewing time der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total viewing time der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die total viewing time der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total viewing time der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total number of fixations	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die total number of fixations der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of fixations der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average number of fixations	Die average number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die average number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die average number of fixations der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average number of fixations der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total fixation duration	Die total fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die total fixation duration der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total fixation duration der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average fixation duration	Die average fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die average fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die average fixation duration der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average fixation duration der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Fixation rate	Die fixation rate der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die fixation rate der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die fixation rate der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die fixation rate der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total number of saccades	Die total number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C

Tabelle 4.2: Auflistung der phasenbasierten Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C

Abhängige Variable	H0	H1
	Die total number of saccades der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of saccades der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average number of saccades	Die average number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die average number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die verage number of saccades der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die verage number of saccades der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total number of visits on erroneous lines	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average number of visits on erroneous lines	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total dwell time on erroneous lines	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average dwell time on erroneous lines	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C
	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad

4.3 Formulierung von Forschungsfragen und Hypothesen zur Identifikation von Visueller Expertise bei der objektorientierten Programmiersprache C++

Die im Kapitel 6 dargestellte Studie widmet sich der Erforschung von visueller Expertise bei der objektorientierten Programmiersprache C++. Ihr Design basiert ebenfalls auf einem kontrastiven Vergleich von Novizen und Experten. Die relevanten Forschungsfragen (siehe Absatz 4.3.1) und die von diesen abgeleiteten Hypothesen (siehe Absatz 4.3.2) werden nachfolgend erläutert.

4.3.1 Forschungsfragen zur Erforschung visueller Expertise bei der objektorientierten Programmiersprache C++

Die Studie zur Erforschung von visueller Expertise bei der objektorientierten Programmiersprache C++ orientiert sich hinsichtlich ihrer Forschungsfragen an der zuvor beschriebenen Studie. Sie verfolgt ebenfalls zwei zentrale Forschungsfragen, welche in ihrer Formulierung lediglich auf einen Wechsel der Programmiersprache abzielen:

- Forschungsfrage 1: *Wie unterscheiden sich Novizinnen und Novizen von Expertinnen und Experten hinsichtlich ihrer Augenbewegungen während eines Code Reviews in der Programmiersprache C++ im Allgemeinen?*

- Forschungsfrage 2: *Welche phasenbasierten Unterschiede machen sich in den Augenbewegungen von Novizinnen und Novizen im Vergleich zu Expertinnen und Experten bei der Durchführung eines Code Reviews in der Programmiersprache C++ deutlich?*

Forschungsfrage 1 zielt erneut auf die Erfassung von allgemeinen Unterschieden hinsichtlich der Augenbewegungen während eines Code Reviews ab. Im Falle dieser Studie wird jedoch die Programmiersprache C++-Quellcodes verwendet. Es werden ebenfalls aufsummierte Eye-Tracking-Metriken bzw. deren Durchschnittswerte als abhängige Variable genutzt. Wie in der vorher beschriebenen Studie werden diese Metriken um weitere Kennzahlen ergänzt welche die Leistung der Versuchspersonen (beispielsweise die Anzahl der gefundenen Fehler oder die Bearbeitungszeit) erfassen.

Forschungsfrage 2 setzt im Falle dieser Studie ebenfalls eine Datenfilterung voraus, welche die Augenbewegungen der Versuchspersonen in verschiedene Phasen unterteilt. Der Vergleich dieser Phasen bedient sich nahezu der gleichen Metriken, welche bereits zur Beantwortung von Forschungsfrage 1 genutzt werden, legt dabei jedoch den Fokus auf Unterschiede im Verlauf des Code Reviews. Diese Vorgehensweise ermöglicht Rückschlüsse auf mögliche Strategiewechsel oder -anpassungen während der Bearbeitung der Codebeispiele.

4.3.2 Hypothesen zur Erforschung visueller Expertise bei der objektorientierten Programmiersprache C

Tabelle 4.3: Auflistung der allgemeinen Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C++

Abhängige Variable	H0	H1
Performance	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht zwischen Novizen und Experten	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten
Total viewing time	Die total viewing time während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total viewing time während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Total number of fixations	Die total number of fixations während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of fixations während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Average number of fixations	Die average number of fixations während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of fixations während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Total fixation duration	Die total fixation duration während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total fixation duration während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Average fixation duration	Die average fixation duration während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average fixation duration während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Fixation rate	Die fixation rate während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die fixation rate während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Total number of saccades	Die total number of saccades während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of saccades während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Average number of saccades	Die average number of saccades während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of saccades während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten

Tabelle 4.3: Auflistung der allgemeinen Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C++

Abhängige Variable	H0	H1
Total number of visits on erroneous lines	Die total number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Average number of visits on erroneous lines	Die average number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Total dwell time on erroneous lines	Die total dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die total dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten
Average dwell time on erroneous lines	Die average dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten	Die average dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten

Für Forschungsfrage 2 ergeben sich im Falle der C++-Studie erneut 12 Hypothesenpaare. Diese decken ebenfalls phasenbasierte Unterschiede zwischen den Erfahrungsstufen der Versuchspersonen ab. Tabelle 4.4 gibt nachfolgend einen Überblick über die zu untersuchenden Hypothesen:

Tabelle 4.4: Auflistung der phasenbasierten Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C++

Abhängige Variable	H0	H1
Total viewing time	Die total viewing time der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total viewing time der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total viewing time der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total viewing time der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total number of fixations	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total number of fixations der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of fixations der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad

Tabelle 4.4: Auflistung der phasenbasierten Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C++

Abhängige Variable	H0	H1
Average number of fixations	Die average number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die average number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die average number of fixations der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average number of fixations der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total fixation duration	Die total fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total fixation duration der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total fixation duration der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average fixation duration	Die average fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die average fixation duration der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die average fixation duration der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average fixation duration der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Fixation rate	Die fixation rate der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die fixation rate der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die fixation rate time der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die fixation rate time der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total number of saccades	Die total number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total number of saccades der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of saccades der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average number of saccades	Die average number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die average number of saccades der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die average number of saccades der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average number of saccades der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad

Tabelle 4.4: Auflistung der phasenbasierten Hypothesen zur visuellen Expertise bei Code Reviews in der Programmiersprache C++

Abhängige Variable	H0	H1
Total number of visits on erroneous lines	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average number of visits on erroneous lines	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average number of visits on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Total dwell time on erroneous lines	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die total total dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad
Average dwell time on erroneous lines	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++ nicht	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++
	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich nicht bezüglich des Expertisegrad	Die average dwell time on erroneous lines der einzelnen Phasen unterscheidet sich bezüglich des Expertisegrad

4.4 Formulierung von Forschungsfragen und Hypothesen zum Aufbau eines Modells zur Interpretation visueller Expertise bei Code Reviews

Im Vergleich zu den vorangegangenen Absätzen, welche sich direkt mit der Durchführung von hypothesengestützten empirischen Studien beschäftigen (sieh Kapitel 5 und Kapitel 6), bedient sich das dritte Ziel dieser Dissertation eher an deren Ergebnissen. Ziel 3 adressiert explizit den Aufbau eines datengetriebenen Modells zur Interpretation von visueller Expertise bei Code Reviews.

Zur Realisierung von Ziel 3 sollen die durch die Studien erlangten Ergebnisse mit den zuvor beschriebenen verschiedenen *holistic models of image perception* (siehe Ab-

satz 2.3 verglichen werden. Die Modelle (Kundel et al., 2007; Nodine & Kundel, 1987; Sheridan & Reingold, 2017; Swensson, 1980), sowie der zuvor behandelte Stand der Forschung (Begel & Vrzakova, 2018; Busjahn et al., 2015; Peachock et al., 2017; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006) legen nahe, dass diese eine solide Grundlage für die Analyse und Interpretation von Code Reviews in verschiedenen Programmiersprachen darstellen könnten.

Speziell zu Ziel 3 formulierte Forschungsfragen fokussieren sich daher auf die Modellbildung. Folgenden Fragen soll nachgegangen werden:

- Forschungsfrage 1: *Welches der holistic models of image perception eignet sich am besten für die Analyse und Interpretation von Augenbewegungen während eines Code Reviews?*
- Forschungsfrage 2: *Welche Komponenten der holistic models of image perception finden sich bei Code Reviews wieder?*
- Forschungsfrage 3: *Wie müsste ein vereinheitlichtes Gesamtmodel zur Analyse und Interpretation von Augenbewegungen während eines Code Reviews aussehen?*

Kapitel 5

Studie 1: Visuelle Expertise bei Code Reviews in der prozeduralen Programmiersprache C

Das nachfolgende Kapitel basiert auf zwei Konferenzbeiträgen. Der erste Beitrag wurde 2018 bei der *IATED International Conference of Education, Research and Innovation (ICERI)* in Sevilla präsentiert und trägt den Titel *"Eye Movements in Software Engineering - What differs the expert from the novice?"*. Dieser befasst sich vorrangig mit der Analyse der grundlegenden Eye-Tracking-Daten und deren Nutzen für die Identifikation von visueller Expertise im Software Engineering (Hauser, Reuter, Hutzler, Mottok & Gruber, 2018). Eine erweiterte phasenbasierte Betrachtung der relevanten Eye-Tracking-Metriken im Kontext der Erforschung visueller Expertise (welche sich so auch im nachfolgenden Kapitel 6 findet) fand in einem zweiten Konferenzbeitrag statt. Dieser trägt den Titel *"Visual expertise in code reviews: Using holistic models of image perception to analyze and interpret eye movements"* und wurde 2023 im Rahmen des von ACM organisierten Workshop *Eye Movements in Programming* in Tübingen vorgestellt und publiziert (Hauser, Grabinger, Mottok & Gruber, 2023).

Kurzfassung

Diese Studie beschäftigt sich mit der Erforschung von visueller Expertise bei der prozeduralen Programmiersprache C unter der Einbeziehung der *holistic models of image perception*. Das Design der Studie basiert auf der Arbeit von Uwano et al. (2006) und Sharif et al. (2012), modifiziert dieses jedoch um Ansätze aus der Expertiseforschung. Ihr stehen die Daten von 23 Versuchspersonen zur Verfügung welche sich in 15 Novizen und acht Experten unterteilen lassen. Im Rahmen des Experiments begutachten die Versuchspersonen sechs Codebeispiele, von denen ein jedes einen logischen Fehler enthält. Während des gesamten Reviews werden die Augenbewegungen von einem SMI 250RED mobile Eye-Tracker aufgezeichnet. Zusätzlich werden über einen Fragebogen demografische Daten erhoben und die aufgezeichneten

Augenbewegungen nach Abschluss des Experiments durch ein stimulusbasiertes Interview verifiziert. Die Ergebnisse legen nahe, dass zwischen den Experten und Novizen Unterschiede bezüglich der Vorgehensweise bei den Code REviews auftreten. Die Vorgehensweise der Experten entspricht den Grundvorstellungen der *holistic models of image perception*, läuft in Phasen ab welche durch die Verwendung bestimmter Strategien (beispielsweise ein Scan, eine Fehlersuche, ...) gekennzeichnet sind. Weiterhin zeigen die Daten auf, dass die Experten eine effizientere Vorgehensweise an den Tag legen und im Vergleich zu den Novizen über bessere Fähigkeiten hinsichtlich der Aufnahme und Verarbeitung von Informationen aus eine C-Quellcode verfügen.

5.1 Design

Die hier dargestellte Studie stellt keine bloße Replika der Arbeiten von Uwano et al. (2006) und von Sharif et al. (2012) dar. Aus den Erkenntnissen der Studie *Developing visual expertise in Software Engineering* (Nivala et al., 2016) und den Arbeiten anderer Autoren (Begel & Vrzakova, 2018; Busjahn et al., 2015; Sharif et al., 2012; Uwano et al., 2006) wurde klar, dass sich das Leseverhalten von Experten und Novizen unterscheidet, bzw. sich in deren Performance bei der Fehlersuche erfahrungsbedingte Unterschiede abzeichnen. Ein Problem des grundlegenden Studiendesigns dieser Arbeit war (Nivala et al., 2016), dass der Forschungsfokus auf die Augenbewegungen gelegt wurde und begleitende Daten lediglich per Fragebogen erhoben wurden. Dies führte bereits bei der Auswertung zu Überlegungen, wie sich das Design dahingehend überarbeiten ließe, so dass auch ein Einblick in die auftretenden kognitiven Prozesse während des Reviews ermöglicht wird.

Mit ähnlichen Problemen hatten bzw. haben auch andere Forscher in diesem Feld zu tun und es wurden immer wieder Ansätze genannt, die bei den Datenerhebungen selbst auf einen sogenannten *think aloud approach* oder auf retrospektive Interviews setzten (Busjahn, Bednarik & Schulte, 2014; Lin et al., 2016).

Beim sogenannten *think aloud approach* werden die Versuchspersonen zu lautem Denken animiert. Sie sollen während des Experiments ihre Vorgehensweise möglichst detailliert schildern und dabei auch auf Kleinigkeiten eingehen. Zu Bedenken ist dabei, dass der *think-aloud*-Ansatz von den Versuchsteilnehmerinnen und -teilnehmern eine gewisse Übung erfordert und in vielen Fällen erst eingeübt werden muss. Gleichzeitig muss auch bedacht werden, dass dieser Ansatz im Regelfall zu einem deutlich gesteigerten *cognitive load* führt. Dies ist insofern problematisch, da ein Code-Review im Allgemeinen bereits als eine sehr komplexe und anspruchsvolle Aufgabe betrachtet werden kann.

Den retrospektiven Interviews liegt der Gedanke zu Grunde, dass diese möglichst direkt im Anschluss an die Datenerhebung durchgeführt werden. Dadurch soll einem möglichen Vergessen oder einer Verfälschung vorgebeugt werden. Im Rahmen der Interviews werden den Versuchspersonen ihre eigenen Augenbewegungen gezeigt. Aufgabe der Teilnehmerinnen und Teilnehmer ist es, diese zu kommentieren, die

eigene Vorgehensweise zu erläutern und Fragen der Versuchsleiter zu beantworten. Die Augenbewegungen können dabei beliebig häufig wiederholt werden.

Unter Berücksichtigung des Aufgabenniveaus und dem Wunsch, den *cognitive load* möglichst gering zu halten, fiel bei der vorliegenden Studie die Entscheidung auf den Einsatz von retrospektiven Interviews. Diese wurden direkt im Anschluss an die Datenerhebung durchgeführt und orientierten sich an einem vorab formulierten Leitfaden.

5.1.1 Forschungsdesign

Das Forschungsdesign dieser Studie basiert auf den Arbeiten von Uwano et al. (2006) und Sharif et al. (2012), baut aber deren Design zu einem kontrastiven Vergleich zwischen Experten und Novizen um (Ericsson et al., 1993; Ericsson & Towne, 2010; Gruber, 2007). Diese beiden Guppen werden hinsichtlich verschiedener abhängiger Variablen gegenübergestellt und auf mögliche Unterschiede hin untersucht. Ergänzend werden die Annahmen der vorab bereits beschriebenen *holistic models of image perception* (2.3) aufgegriffen. Diese lassen bei der Betrachtung eines visuellen Stimulus auf ein phasenbasiertes Vorgehen schließen (Kundel et al., 2007; Nodine & Kundel, 1987; Swensson, 1980; Sheridan & Reingold, 2017). Die dabei entstehenden Phasen werden sowohl innerhalb der Gruppe, als auch übergreifend über diese betrachtet.

5.1.2 Relevante Variablen und Metriken

Im Falle der vorliegenden Studie basieren die abhängigen Variablen auf der Arbeit von Sheridan und Reingold (2017,S.5). In dieser Studie stellt das Autorenpaar die gemeinsamen Metriken der verschiedenen *holistic models of image perception* vor, welche im Rahmen dieser Dissertation für die Analyse von Code Reviews angewandt werden. Besondere Aufmerksamkeit kommt dabei den fixationsbasierten Metriken zu, welche mit der Aufnahme und Verarbeitung von Informationen assoziiert werden (Duchowski, 2017; Holmqvist & Andersson, 2017; Sheridan & Reingold, 2017). Konkret untersucht werden die folgenden Variablen:

- *Anzahl der gefundenen Fehler*
- *Total number of fixations*
- *Average number of fixations*
- *Total fixation duration*
- *Average fixation duration*
- *Fixation rate*
- *Total number of saccades*

- *Average number of saccades*
- *Total number of visits on erroneous lines*
- *Average number of visits on erroneous lines*
- *Total dwell time on erroneous lines*
- *Average dwell time on erroneous lines*

Als unabhängige Variablen werden die *allgemeine Programmiererfarhung* die *berufliche Programmiererfahrung* genutzt, welche im Sinne der Arbeiten von Ericsson (Ericsson et al., 1993; Ericsson & Towne, 2010; Ericsson, 2016) den besten Gradmesser für die Expertise der Versuchspersonen darstellt.

5.2 Methoden

Die nachfolgenden Absätze geben einen Überblick über die Methodologie dieser Studie. Absatz 5.2.1 gibt einen Überblick über die im Experiment verwendeten Instrumente. Im Anschluss daran wird in Absatz 5.2.2 auf die Zusammensetzung der untersuchten Stichprobe eingegangen, bevor sich Absatz 5.2.3 der Durchführung des Experiments widmet.

5.2.1 Instrumente

Hinsichtlich der verwendeten Instrumente lässt sich diese Studie in vier Bereiche unterteilen. Es soll nachfolgend geschildert werden, welche Codebeispiele als visuelle Stimuli verwendet wurden und welche Fehler diese beinhaltet haben. Weiterhin wird geklärt, welcher Eye-Tracker mit welcher Experimentalsoftware genutzt wurde. Ebenso wird geschildert, welchen Fragebogen die Versuchspersonen ausfüllen mussten und wie sich die nach dem Experiment durchgeführten Interviews gestalteten bzw. wie der Leitfaden für diese gestaltet war.

Codebeispiele als visuelle Stimuli

In dieser Studie werden insgesamt sechs fehlerhafte Quellcodes in der Programmiersprache C als visuelle Stimuli eingesetzt. Diese basieren auf der Arbeit von Uwano et al. (2006). Die verwendeten Codebeispiele werden von der Forschergruppe zwar nicht im Original zur Verfügung gestellt, jedoch beinhaltet die Publikation sehr detaillierte Beschreibungen der Codes und den jeweils enthaltenen Fehlern. Diese Beschreibungen werden dazu genutzt, um die Stimuli möglichst detailliert zu rekonstruieren und für eine Replikation nutzbar zu machen.

In der Arbeit von Uwano et al. (2006) werden ausschließlich logische Fehler eingesetzt. Diese sind für einen Programmierer oder eine Programmiererin nicht auf den ersten Blick sichtbar (wie dies beispielsweise bei Syntax- und Semantikfehlern der Fall sein kann) und erfordern ein gewisses Verständnis des gezeigten Codes.

Dieses Verständnis sollte dabei soweit gehen, dass die Reviewer die Funktion des Codes erfassen und dessen grundlegende Funktion beschreiben können. Die Forschergruppe um Uwano (2006) setzt bei ihrer Studie vorrangig auf Codes, die sich mit mathematischen Rechenoperationen beschäftigen. Folgende Beispiele finden sich in der ursprünglichen Arbeit wieder (eine Originaldarstellung der Stimuli findet sich im digitalen Anhang dieser Dissertation):

Accumulate (LOC=19): Bei dieser Aufgabe soll der Nutzer oder die Nutzerin eine nichtnegative Zahl *n* eingeben. Das Programm gibt im Anschluss die Summe aller ganzen Zahlen von 1 bis *n* an. Fehlerhaft ist an diesem Beispiel allerdings, dass die Bedingung für eine Schleife verletzt wird. Diese Bedingung sollte eigentlich (i <= n) lauten, ist aber stattdessen als (i < n) formuliert. Dieses Beispiel wird insgesamt mit 19 Codezeilen abgedeckt.

```
01.#include <stdio.h>
02.
03.int main()
04.{
05.    unsigned int n;
06.    unsigned int sum = 0;
07.    unsigned int i;
08.
09.    printf("Please enter a non-negative integer value!\n");
10.    scanf("%u", &n);
11.
12.    for(i = 1; i < n; i++) {
13.       sum += i;
14.    }
15.
16.    printf("The sum of all values from 1 to %u is %u.\n", n, sum);
17.
18.    return 0;
19.}
```

Listing 5.1: Codebeispiel zu *Accumulate*

Average-5: In diesem Beispiel sollen die Nutzerinnen und Nutzer fünf ganze Zahlen eingeben. Das Programm soll deren Mittelwert berechnen. Da jedoch eine Typumwandlung (von integer zu double) vergessen wurde, gibt das Program stattdessen einen gerundeten Mittelwert an. Bei dieser Aufgabe gilt es 16 Zeilen Code zu reviewen.

```
01.#include <stdio.h>
02.
03.int main()
04.{
05.   int a=0, b=0, c=0, d=0, e=0;
06.
07.   printf("Please enter 5 integer values separated with spaces!\n");
08.   scanf("%d %d %d %d %d", &a, &b, &c, &d, &e);
09.
10.   int sum = a + b + c + d + e;
11.   int average = sum / 5;
12.
```

```
13.    printf("The average of all entered values is %d.\n", average);
14.
15.    return 0;
16.}
```

Listing 5.2: Codebeispiel zu *Average-5*

Average-any: Bei dieser Aufgabe können die Nutzerinnen und Nutzer eine beliebige Anzahl ganzer Zahlen (bis zu 255) eingeben, bis die Zahl Null erreicht wird. Das dargestellte Programm gibt den Mittelwert der eingegebenen Zahlen aus. Fehlerhaft ist in diesem Fall die Anzahl der Schleifen. Das Programm geht immer davon aus, dass 255 Zahlen eingegeben wurden, unabhängig von der tatsächlichen Anzahl der Eingaben. Dieses Beispiel umfasst 22 Zeilen Code.

```
01.#include <stdio.h>
02.
03.int main()
04.{
05.    int numbers[255];
06.    int number = 0;
07.    int scanCount = 0, indexSum = 0;
08.    double sum = 0;
09.
10.    printf("Please enter integer values to get the average. Enter 0 to finish.\n");
11.
12.    do
13.    {
14.        number = 0;
15.        scanf("%d", &number);
16.        if(scanCount < 255 && number != 0)
17.        {
18.            numbers[scanCount] = number;
19.            scanCount++;
20.        }
21.    }
22.    while (scanCount < 255 && number != 0);
23.
24.    for (indexSum = 0; indexSum < 255; indexSum++)
25.    {
26.        sum += numbers[indexSum];
27.        printf("Summe: %f\n", sum);
28.    }
29.
30.    double average = sum / scanCount;
31.
32.    printf("The average of all entered values is %f.\n", average);
33.    return 0;
34.}
```

Listing 5.3: Codebeispiel zu *Average-any*

Prime: Dieses Programm soll überprüfen, ob es sich bei einer ganzzahligen Eingabe um eine Primzahl handelt oder nicht. Allerdings wird dabei fälschlicherweise die Logik eines bedingten Ausdrucks umgedreht, was zu einem gegenteiligen Ergebnis führt. Um diese Funktion darzustellen, werden 18 Zeilen Code benötigt.

```
01.#include <stdio.h>
02.#include <math.h>
03.
04.int main()
05.{
06.    int n=0;
07.    int i;
08.    int prime = 0;
09.
10.    printf("Please enter an integer value\n");
11.    scanf("%d", &n);
12.
13.    if(n > 1)
14.    {
15.        for(i = 2; i <= sqrt(n); i++)
16.        {
17.            if(n%i != 0)
18.            {
19.                prime = 0;
20.                break;
21.            }
22.            prime = 1;
23.        }
24.    }
25.
26.    printf("number is %sprime.\n", prime == 1 ? "" : "no ");
27.    return 0;
28.}
```

Listing 5.4: Codebeispiel zu *Prime*

Sum-5: Bei diesem Beispiel soll der Benutzer oder die Benutzerin fünf ganze Zahlen eingeben. Das Programm soll deren Summe ausgeben. Der Fehler liegt darin, dass die Variable, die die Summe bilden soll nicht initialisiert wird. Insgesamt umfasst diese Aufgabe 12 Zeilen Code.

```
01.#include <stdio.h>
02.
03.int main()
04.{
05.    int a=0, b=0, c=0, d=0, e=0;
06.
07.    printf("Please enter 5 integer values separated with spaces!\n");
08.    scanf("%d %d %d %d %d", &a, &b, &c, &d, &e);
09.
10.    sum = a + b + c + d + e;
11.
12.    printf("The sum of all entered values is %d.\n", sum);
13.
14.    return 0;
15.}
```

Listing 5.5: Codebeispiel zum *Sum-5*

Swap: Dieses Programm fordert die Nutzerinnen und Nutzer dazu auf, zwei Zahlen einzugeben. Diese Eingaben werden anschließend mit Hilfe der Funktion swap() vertauscht. Darauf folgend gibt das Programm das Ergebnis aus. Der Fehler liegt

bei diesem Problem in einer fehlerhaften Verwendung der Pointer, die zur Folge hat, dass die beiden Zahlen nicht vertauscht werden. Dieses Beispiel basiert auf 23 Zeilen Code.

```
01.#include <stdio.h>
02.
03.int main()
04.{
05.    int i = 0;
06.    int j = 0;
07.
08.    printf("Please enter one integer value\n");
09.    scanf("%d", &i);
10.
11.    printf("Please enter another integer value\n");
12.    scanf("%d", &j);
13.
14.    printf("First number: %d, second number: %d\n", i,j);
15.
16.    swap(i, j);
17.    printf("After swap:\nFirst number: %d, second number: %d\n", i,j);
18.
19.    return 0;
20.}
21.
22.void swap(int i, int j)
23.{
24.    int temp = i;
25.    i = j;
26.    j = temp;
27.}
```

Listing 5.6: Codebeispiel zu *Swap*

Die Codebeispiele werden anlässlich der Studie von einer Masterstudentin und Doktorandin der Informatik erstellt und anschließend von mehreren Reviewern hinsichtlich ihrer Eignung überprüft. Durch diese Vorgehensweise soll sichergestellt werden, dass die enthaltenen Fehler den Beschreibungen von Uwano et al. (2006) entsprechen und keine weiteren Fehler oder Ungenauigkeiten im Code enthalten sind.
Darüber hinaus wird bei der Erstellung der Stimuli darauf geachtet, dass diese in einer möglichst neutralen Form präsentiert werden können. Daher wird auf Highlightingoptionen verzichtet. Der Code wird in der Schriftart "Courier New" mit einer Schriftgröße von 14pt als schwarzer Text auf einem weißen Hintergrund dargestellt.
In Abstimmung mit den technischen Gegebenheiten des Eye-Trackers (SMI 250 REDmobile) wird zusätzlich darauf geachtet, dass die Stimuli möglichst mittig auf dem verwendeten Screen präsentiert werden. Dieser Bereich lässt sich für die meisten geläufigen Eye-Tracker bestmöglich kalibrieren, womit eine möglichst genaue Messung der gezeigten Augenbewegungen gewährleistet werden kann (iMotions, 2022; SensoMotoric Instruments, 2017; Tobii, 2010; Tobii, 2020).

Verwendeter Eye-Tracker

Die Aufzeichnung der Augenbewegungen erfolgte im Falle dieser Studie mit einem Eye-Tracker des Typs SMI 250 REDmobile. Dieser verfügt über eine sampling rate von bis zu 250Hz, welche stufenweise auf 120Hz oder 60Hz abgesenkt werden kann. Seine accuracy wird mit einem Wert von 0,4° angegeben (iMotions, 2022). Beide Angaben bewegen sich in einem für die Untersuchung von Quellcode angemessenen Rahmen (Sharafi et al., 2020). Durch eine sampling rate von mehr als 120Hz können auch Saccaden erfasst werden und eine accuracy von <0,5° ist für Codebeispiele geeignet. Der verwendete Eye-Tracker wurde an einem externen 24”-Monitor befestigt. Dies hat mehrere Gründe. Durch den größeren Monitor wird eine bessere Darstellung der Quellcodes ermöglicht, was sich weiterhin auch positiv auf den Komfort der Versuchsteilnehmerinnen und -teilnehmer auswirkt Gleichzeitig werden Vibrationen bei der Eingabe der Antworten reduziert, da ergänzend noch eine externe Tastatur verwendet wird und nicht auf einem Laptop geschrieben wird. 24” stellen darüber hinaus auch das Maximum an des erfassbaren Bereiches des SMI 250 REDmobile dar (iMotions, 2022; SensoMotoric Instruments, 2017). Als Experimentalsoftware wurde die werksseitig zur Verfügung gestellte Toolsuite von SensoMotoric Instruments (2017) genutzt. Diese setzt sich aus einer Experimentalsoftware (iView) und einer Analysesoftware (BeGaze) zusammen. Durch den Aufkauf von SensoMotoric Instruments durch Apple und den damit verbundenen Wegfall des Supports für Hard- und Software handelt es sich bei der vorliegenden Version 3.7 um den finalen Stand des Programms. Das Experiment selbst wird in iView aufgesetzt. Es werden zu Beginn standardisierte Folien gezeigt, welche den Versuchspersonen entsprechende Arbeitsanweisungen geben. Die Stimuli selbst werden als Bilddateien (.jpg) in das Tool gezogen. Die Codebeispiele sind statisch und können von den Versuchsteilnehmerinnen und -teilnehmern nicht edi- tiert werden. Zur Beschreibung der Fehler wird nach jedem Codebeispiel eine Folie mit einem Textfeld platziert, auf dem die Versuchspersonen ihre Antwort formulieren können. Sowohl die Blickbewegungsdaten, als auch die Antworten werden während des gesamten Experiments erfasst und von iView entsprechend abgelegt (SensoMotoric Instruments, 2017). Für die Aufbereitung und grundlegende Analyse der Eye-Tracking-Daten wird SMI BeGaze eingesetzt (SensoMotoric Instruments, 2017). Bei dieser Software handelt es sich um die standardmäßige Auswertungssoftware des Herstellers des eingesetzten Eye- Trackers und bietet ein breites Anwendungsspektrum. Dieses deckt alle von Sharafi et al. (2020, S.3135-3139) beschriebenen Phasen der Auswertung einer Eye-Tracking-Studie im Software Engineering ab. Der Fokus liegt hier jedoch vorwiegend auf der Datenauf- bereitung und die Erfassung von AOIs durch BeGaze (SensoMotoric Instruments, 2017). Weiterführende Eye-Tracking-Metriken werden händisch auf Grundlage der vorgefilterten Daten in R errechnet und analysiert.

Qualitativer Fragebogen

Der eingesetzte Fragebogen unterteilt sich in mehrere Teilbereiche. Er beinhaltet Hintergrundinformationen zur Studie und eine Einwilligungserklärung, welcher die Versuchs- personen vor Beginn des Experiments zustimmen müssen. Vorrangig dient er jedoch dazu, demografische Daten und Selbsteinschätzungen der Teilnehmerinnen und Teilnehmer zu erheben, welche im Kontext des kontrastiven Vergleichs zwischen den verschiedenen Erfahrungsstufen engesetzt werden können. Bezüglich der demografischen Daten soll angegeben werden, über wie viele Jahre allgemeine, sowie über berufliche Programmiererfah- rung verfügt wird. Weiterhin wird abgefragt, welchem Versuch die teilnehmende Person nachgeht und welche aus Studiensicht relevanten Zusatzkenntnisse (z.B. Zertifizierung zum Softwaretester) vorhanden sind.

Interviewleitfaden

Im Sinne der methodischen Triangulation kam bei dieser Studie neben dem qualitativen Fragebogen ein stimulusbasiertes Interview als ergänzende Erhebungsmethode zum Einsatz. Im Rahmen dieses Interviews wurde den Versuchspersonen die Aufzeichnung ihrer eigenen Augenbewegungen gezeigt. Dabei hatten sie die Möglichkeit diese zu kommentieren und ihr Vorgehen näher zu erläutern. Zur Durchführung der Interviews wurde ein offener Interwieleitfaden erstellt. Dieser enthielt vorab formulierte Fragen, die bei jedem Interview gestellt wurden. Konkret handelte es sich dabei um die folgenden Fragen:

- ”Wie haben Sie das Experiment empfunden?”
- ”Wie haben Sie die Codebeispiele eingeschätzt? Fanden Sie diese eher leicht oder schwierig?”
- ”Hatten Sie bei Ihren Reviews eine bestimmte Strategie?”

Die genaue Formulierung der Fragen konnte bei der Durchführung des Interviews leicht variiert werden, jedoch maximal in einem Rahmen, welcher den dahinterstehenden Sinn nicht verzerrte. Ebenso sollte während der Interviews auf Auffälligkeiten eingegangen werden. So sollte geklärt werden, warum beispielsweise eine bestimmte Stelle eine sehr lange *fixation duration* erhielt oder es zu häufigen Transitionen zwischen bestimmten Codesegmenten kam. In diesem Fall sah der Leitfaden vor, dass diese Auffälligkeiten gezielt hinterfragt wurden. Entsprechende Fragen waren vorab nicht formuliert, konnten aber folgendermaßen lauten:

- ”Sie haben sich recht lange auf diese Zeile des Codes konzentriert. Ist Ihnen dort etwas aufgefallen?”
- ”Sie sind anfangs eher durch den Code gesprungen. Hatte das einen bestimmten Grund?”

Durch die stimulusbasierten Interviews sollten vorrangig zwei Ziele abgedeckt werden: Die im Experiment aufgezeichneten Augenbewegungen liefern zwar eine Vielzahl an Informationen, deren Kontext ist jedoch nicht unbedingt klar ersichtlich bzw. nachvollziehbar. Daher bedürfen diese und die darauf basierenden Daten einer entsprechenden Validierung. So soll sichergestellt werden, dass störende Einflüsse (z.B. Ablenkung der Versuchsperson, Kalibrierungsprobleme des Eye-Trackers, Überforderung, Desinteresse, ...) erkannt und für den weiteren Verlauf der Datenauswertung berücksichtigt werden. Die stimulusbasierten Interviews zielten darauf ab, dass die Versuchspersonen ihre Strategien und Vorgehensweisen bei den Code Reviews erklärten. Durch die Gegenüberstellung der Erklärungen und der Augenbewegungen sollen diese erklärt werden und Unterschiede zwischen Experten und Novizen transparent gemacht werden.

5.2.2 Stichprobe

An dieser Studie nimmt eine Stichprobe von insgesamt 25 Versuchspersonen teil. Es gilt jedoch zu beachten, dass die Datensätze von zwei Probanden verworfen werden mussten. Bei beiden Personen handelte es sich um sehr helläugige Menschen, in deren Fall der verwendete Eye-Tracker für eine reliable Messung nicht kalibriert werden konnte. Somit stehen für die weitere Analyse der Ergebnisse die Datensätze von 23 Versuchsteilnehmerinnen und -teilnehmern zur Verfügung.
Die Stichprobe kann weiterhin in Fortgeschrittene/Experten und Novizen unterteilt werden. Die Unterteilung in entsprechende Erfahrungsstufen basiert im Falle dieser Studie auf dem demographischen Merkmal der allgemeinen Programmiererfahrung. Die Gruppen können folgendermaßen beschrieben werden:

- *Novizen:* Als Novize wird im Kontext dieser Studie eine Person bezeichnet, die über weniger als fünf Jahre allgemeine Programmiererfahrung verfügt. Dennoch verfügen die Mitglieder dieser Gruppe über Programmierkenntnisse in der Sprache C, die für die Bearbeitung der konzipierten Codebeispiele ausreichend ist. Bei den Versuchspersonen dieser Gruppe handelt es sich im Fall der vorliegenden Studie um Studierende aus informatiklastigen Bachelorstudiengängen der OTH Regensburg, die in entsprechenden Lehrveranstaltungen rekrutiert worden sind.

- *Fortgeschrittene/Experten:* Diese Gruppe verfügt über mindestens fünf Jahre allgemeine Programmiererfahrung und kann auch auf professionelle Tätigkeiten im Bereich der Informatik zurückblicken. Die Versuchspersonen dieser Gruppe sind mit der Progarmmiersprache C vertraut und arbeiten regelmäßig mit dieser in ihrem Berufsalltag. Bei den Probandinnen und Probanden dieser Erfahrungsstufe handelt es sich um Doktorandinnen und Doktoranden der Informatik, sowie um professionelle Entwicklerinnen und Entwickler aus kooperierenden Unternehmen.

Bezüglich der Rekrutierung von Versuchspersonen gilt es im Bereich des Software

Tabelle 5.1: Demografische Angaben zur gesamten Stichprobe (N=23)

Variable	**Mean**	**SD**	**Min**	**Max**
Alter	24.174	2.674	20.000	29.000
Allgemeine Programmiererfahrung	4.500	2.671	2.000	10.000
Professionelle Programmiererfahrung	.848	1.393	.000	5.000

Tabelle 5.2: Demographische Angaben zur Expertengruppe (n=8)

Variable	**Mean**	**SD**	**Min**	**Max**
Alter	26.375	1.923	23.000	28.000
Allgemeine Programmiererfahrung	7.625	2.066	5.000	10.000
Professionelle Programmiererfahrung	2.000	1.604	.000	5.000

Engineerings anzumerken, dass sich diese im Regelfall schwieriger gestaltet. In den meisten Fällen müssen die teilnehmenden Personen bereits über bestimmtes Wissen oder spezielle Fähigkeiten verfügen (beispielsweise grundlegende Programmierkenntnisse in einer für das Studiendesign relevanten Sprache). Insbesondere trifft dies auf Expertinnen und Experten zu (Sharif & Mansoor, 2022). Bei diesen handelt es sich meist um guteingebundene Programmiererinnen und Programmierer, welche essentieller Bestandteile ihrer Teams sind und eigenverantwortlich für die Erstellung umfangreicher Codes verantwortlich sind. Eine Versuchsteilnahme während der regulären Arbeitszeiten erweist sich somit als verhältnismäßig schwierig.
Ebenso zeichnen sich auch bei der Rekrutierung von Studierenden Probleme ab. In deren Fall muss sichergestellt sein, dass die relevanten Grundkenntnisse bereits vorhanden sind und eine Versuchsteilnahme mit dem individuellen Vorlesungszeit vereinbar ist. Somit ist eine relativ willkürliche Rekrutierung in der Domäne des Software Engineerings im Normalfall nicht möglich (Sharif & Mansoor, 2022). Für den Standort Regensburg ergab sich zum Zeitpunkt der Datenerhebung das Problem, dass die Universität Regensburg noch über keine eigene Fakultät für Informatik oder eine studentische Programmierausbildung verfügte, in deren Rahmen das relevante Wissen für diese Studie vermittelt wird. Somit blieb die Rekrutierung auf die informatiklastigen Studiengänge der OTH Regensburg beschränkt.
Eine genauere Beschreibung der Stichprobe kann den nachfolgenden Tabellen entnommen werden. Diese behandeln die Stichprobe im Allgemeinen und separieren diese auch in die beiden Erfahrungsgruppen:
Zusammenfassend zeigt sich, dass mit einem Durchschnittsalter von 24.174

Tabelle 5.3: Demographische Angaben zur Novizengruppe (n=15)

Variable	**Mean**	**SD**	**Min**	**Max**
Alter	23.000	2.268	20.000	29.000
Allgemeine Programmiererfahrung	2.833	.724	2.000	4.000
Professionelle Programmiererfahrung	.233	.776	.000	3.000

(SD=2.674) Jahren eine verhältnismäßig junge Stichprobe in diesem Bereich vorliegt. Dies ist vor allem darauf zurückzuführen, dass lediglich eine kleine Anzahl von Fortgeschrittenen und Experten (n=8) rekrutierte werden konnte. Diese geringe Probandenanzahl hat auch zur Folge, dass keine herausgelöste Zwischengruppe von Fortgeschrittenen abgebildet werden kann und diese zusammengefasst mit den Experten betrachtet werden müssen.

5.2.3 Durchführung

Die Rekrutierung der Stichprobe für dieses Experiment erfolgte während des Lehrbetriebs an der OTH und Universität Regensburg. Studierende (welche in diesem Falle die Novizen-Gruppe darstellten) aus thematisch relevanten Vorlesungen und Seminaren aus der Informatik und Elektro- und Informationstechnik wurden auf die Teilnahmemöglichkeit hingewiesen und auf freiwilliger Basis rekrutiert. Falls möglich, konnten diese die Versuchsteilnahme mit zu absolvierenden Pflichtübungen verrechnen. Die Anwerbung von Fortgeschrittenen bzw. Experten stellte sich als schwieriger dar. Im Falle der hier vorgestellten Studien handelt es sich bei diesen meist um Doktoranden aus den Computerwissenschaften, welche über bestehende Kontakte seitens des LaS^3 angeworben werden konnten.
Die Datenerhebung fand in einem geeigneten Büro an der OTH Regensburg statt. Dieses war mit Jalousien ausgestattet und wurde von Leuchtstoffröhren beleuchtet. Aufgrund der eher abgelegenen Lage auf dem Campusgelände und der Kontrollierbarkeit der äußeren Einflüsse kann von einem Laborsetting gesprochen werden.
Das Experiment unterteilte sich in vier Phasen:

1. *Aufklärungsgespräch und Instruktion:* Vor der Datenerhebung wurden die Versuchspersonen über die Studie und ihre Ziele aufgeklärt und gaben ihr Einverständnis für die Teilnahme. Eine genauere Instruktion der Versuchspersonen erfolgte nach deren Zustimmung.

2. *Ausfüllen des Fragebogens:* Die Versuchspersonen wurden gebeten, den zuvor genannten Fragebogen auszufüllen. Dieser lag ihnen in Papierform vor.

3. *Erhebung der Eye-Tracking-Daten:* Das Eye-Tracking-Experiment wurde mit SMI BeGaze (SensoMotoric Instruments, 2017) aufgesetzt und durchgeführt. Vor jedem Codebeispiel erfolgte eine Kalibrierung des Eye-Trackers um eine größtmögliche Genauigkeit gewährleisten zu können (Holmqvist et al., 2011). Nach jedem Codebeispiel gab es die Möglichkeit auf einer Folie eine Freitextantwort zu formulieren. Die Augenbewegungen der Versuchspersonen wurden auch in dieser Phase aufgezeichnet

4. *Durchführung eines stimulusbasierten Interviews:* Direkt im Anschluss an die Erhebung der Eye-Tracking-Daten wurde ein stimulusbasiertes Interview mit den Versuchspersonen durchgeführt. In dessen Rahmen konnten die Teilnehmerinnen und Teilnehmer ihre eigenen Augenbewegungen als Video sehen. Auf-

grund der zeitnahen Durchführung sollte sichergestellt werden, dass es zu keinen Verfälschungen auf Seiten der Versuchspersonen kommt. Sie wurden dazu animiert diese zu kommentieren und ihre Vorgehensweise zu erklären. Parallel dazu wurde in dieser Phase des Experiments der zuvor formulierte Interviewleitfaden eingesetzt. Die gesamten Interviews wurden als Audiomitschnitt aufgezeichnet. Auffälligkeiten wurden seitens des Experimentalleiters angesprochen und notiert. Den Teilnehmern stand es weiterhin aufgrund der relativ langen Experimentaldauer frei, die Aufzeichnung zu überspringen oder diese nur teilweise anzusehen.

Die Gesamtdauer des Experimentes variierte stark und war abhängig von der jeweiligen Versuchsperson bzw. den relevanten Einflussfaktoren (z.B. Expertisegrad). Die Spannweite erstreckt sich von ca. 20 Minuten bis hin zu über 45 Minuten.

5.3 Datenauswertung

Die Analyse der im Experiment gesammelten Daten unterteilt sich insgesamt in vier Abschnitte. Der erste Schritt beschäftigt sich damit, die Daten aufzubereiten und für die weitere Analyse in ein geeignetes Format zu bringen (siehe Absatz 5.3.1). Darauf folgend werden die demographischen Daten der Versuchspersonen, sowie zusätzliche Informationen (z.B. erzielte Punkte im Experiment, ...) zu diesen analysiert (siehe Absatz 5.3.2). Anschließend wird eine Analyse der grundlegenden Eye-Tracking-Daten präsentiert, in deren Rahmen auf Auffälligkeiten zwischen den einzelnen Erfahrungsstufen eingegangen wird (siehe Absatz 5.3.3). Abgeschlossen wird die Datenauswertung durch die phasenbasierte Betrachtung der Eye-Tracking-Daten, welche in Absatz 5.3.4 dargestellt wird.

5.3.1 Datenaufbereitung

Bevor mit den Daten des Experiments gearbeitet werden konnte, mussten diese gesammelt und in ein auswertbares Format gebracht werden. Abhängig vom jeweiligen Erhebungsinstrument waren dabei verschiedene Schritte der Datenaufbereitung notwendig. Die Vorgehensweise lief folgendermaßen ab:

Fragebogen: Die im Fragebogen angegebenen demographischen Daten, sowie die Selbsteinschätzung wurden individuell für jede Versuchsperson in ein maschinenlesbares Format gebracht. Diese Daten konnten in einem gemeinsamen Datensatz mit den verschiedenen Eye-Tracking-Metriken kombiniert werden. Die Angaben der Versuchspersonen wurden dabei exakt übernommen.

Eye-Tracking-Daten: Die Aufbereitung der Eye-Tracking-Daten orientierte sich an der von Sharafi et al. (2020, S.3135-3139) beschriebenen vierstufigen Vorgehensweise und griff auf eine Kombination aus verschiedenen Softwareprodukten zurück. Der

Ablauf der Datenfilterung wird nachfolgend in Anlehnung an Sharafi et al. (2020, S. 3135-3139) beschrieben:

1. *Ungefilterte Rohdaten (First Order Data):* Die SMI-Software BeGaze (SensoMotoric Instruments, 2017) erlaubte einen breiten Zugriff auf alle relevanten Daten. Die Rohdaten des Eye-Trackers konnten erfolgreich abgegriffen werden.

2. *Gefilterte Rohdaten (Second Order Data):* SMI BeGaze bietet verschiedene Filteroptionen zur Ermittlung von *fixations* und *saccades*. Im Falle der vorliegenden Studie wurden die Standardfilter von SMI benutzt (SensoMotoric Instruments, 2017). Ergänzende Filter (beispielsweise über R-Packages) waren nicht notwendig. Die dabei erlangten Daten wurden dabei im weiteren Verlauf der Datenauswertung teilweise für die Berechnung von *komplexen Eye-Tracking-Metriken (Fourth Order Data)* verwendet.

3. *Grundlegende Eye-Tracking-Metriken (Third Order Data):* SMI BeGaze bietet standardmäßig bereits eine große Anzahl von grundlegenden Eye-Tracking-Metriken an (SensoMotoric Instruments, 2017). Diese können nach Abschluss des Datenexports direkt verwendet werden. Die entsprechenden Metriken wurden vor ihrem Export hinsichtlich ihrer Relevanz für die vorab formulierten Hypothesen überprüft und mit Standardwerken des Eye-Trackings (Duchowski, 2017; Holmqvist et al., 2011; Holmqvist & Andersson, 2017) abgeglichen. Dieser Abgleich bedingte sich durch die geräte- bzw. herstellerspezifische Definition der jeweiligen Metriken (SensoMotoric Instruments, 2017; Tobii Pro, 2021) und hatte das Ziel, diese mit der nachfolgenden Studie (Hauser, Schreistetter et al., 2020) vergleichbar zu machen, welche einen Tobii Pro Spectrum (Tobii Pro, 2020) genutzt hat.

4. *Komplexe Eye-Tracking-Metriken (Fourth Order Data):* Die Berechnung der komplexen Eye-Tracking-Metriken erfolgte auf Basis der *Second* und *Third Order Data* und wurde in R durchgeführt. Im Falle der vorliegenden Studie spielten in Bezug auf diese Kategorie vorrangig Verhältnisse zwischen einzelnen Variablen (z.B. *fixation rate, proportion of dwell time on erroneous lines, ...*), sowie phasenbasierte Metriken (z.B. *number of fixations during phase 1, ...*) eine entscheidende Rolle. Diese wurden für jedes zu reviewende Codebeispiel einzeln und kumuliert über das gesamte Experiment hinweg berechnet. Die entsprechenden R-Skripte zur Filterung und Berechnung finden sich im digitalen Anhang dieser Dissertation.

Interview: Die Interviewdaten der Versuchspersonen, bei denen Angaben zur Vorgehensweise und zu den verwendeten Strategien gemacht wurden, dienten zur Absicherung der Eye-Tracking-Daten. Sie wurden nicht transkribiert, jedoch als Audioaufzeichnung in die weitere Analyse einbezogen.

5.3.2 Analyse der demographischen Daten

Die Analyse der demographischen Daten zeigt nachfolgend auf, welche Verbindungen sich zwischen den im Experiment erzielten Punkten und der angesammelten Erfahrung der Versuchspersonen herstellen lässt. Weiterhin wird thematisiert, wie sich die Selbsteinschätzung der Versuchsteilnehmerinnen und -teilnehmer darstellt.

Überblick über die im Experiment erzielten Punkte

Tabelle 5.4: Deskriptive Statistik zur Anzahl der im Experiment erzielten Punkte

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	2.652	1.301	2.000	0.000	6.000	6.000
Experten	3.625	1.408	3.500	2.000	6.000	4.000
Novizen	2.133	0.915	2.000	0.000	4.000	4.000

Betrachtet man die gesamte Anzahl der im Experiment erzielten Punkte (siehe Tabelle 5.4, so zeigen sich zwischen den beiden Erfahrungsstufen mehrere Auffälligkeiten. Die Gruppe der Novizen erzielte einen Mittelwert von 2.133 Punkten (SD=0.915; Mdn=2.000). Keine zu dieser Erfahrungsgruppe gehörende Person war in der Lage alle gestellten Aufgaben richtig zu beantworten, es wurde lediglich bei den besten Novizen ein Maximalwert von 4.000 Punkten erzielt. In der Gruppe der Experten gab es im Vergleich dazu Personen, welche in der Lage waren alle Codebeispiele korrekt zu begutachten. Generell zeigt sich, dass die Experten eine bessere Leistung hinsichtlich der Fehlererkennung demonstrieren. Die Gruppe weist einen Mittelwert von 3.625 (SD=1.408; Mdn=3.500) erreichten Punkten auf.
Zur Überprüfung, ob zwischen den beiden Gruppen hinsichtlich der Anzahl der erzielten Punkte ein statistisch signifikanter Unterschied vorliegt wurde ein t-Test durchgeführt. Dieser kann bestätigen, dass sich die Experten signifikant von den Novizen unterscheiden (t(10.252)=-2.707; p=0.022; d=1.371). Basierend auf Cohen (1992) kann dieser Effekt als groß betrachtet werden.
Weithin können zwischen der *Anzahl der erzielten Punkte* und der *allgemeinen* (r_{BP}=0.556; p=0.006), sowie der *professionellen Programmiererfahrung* (r_{BP}=0.659; p=0.000) statistisch signifikante Korrelationen beobachtet werden. Diese deuten darauf hin, dass die individuelle Erfahrung der jeweiligen Versuchsperson einen deutlichen Einfluss auf die erfolgreiche Durchführung eines Code Reviews ausübt. Umso erfahrener ein Reviewer oder eine Reviewerin ist, umso wahrscheinlicher ist es, dass Fehler erkannt werden.
Die in Tabelle 5.5 dargestellte gruppenübergreifende Betrachtung der einzelnen Aufgaben des Experiments zeigt, dass die Versuchspersonen bei den Aufgaben *Average-any* (M=0.304; SD=0.470) und *Sum-5* (M=0.304; SD=0.470) relativ schlechte Ergebnisse erzielten. Umgekehrt verhält es sich mit der Aufgabe *Swap*. Diese schneidet mit einem Mittelwert von 0.739 (SD=0.449) am besten ab.

Tabelle 5.5: Gruppenübergreifende Betrachtung der Aufgaben

Aufgabe	Mittelwert	SD	Mdn
Accumulate	0.470	0.511	0.000
Average-5	0.348	0.487	0.000
Average-any	0.304	0.470	0.000
Prime	0.478	0.511	0.000
Sum-5	0.304	0.470	0.000
Swap	0.739	0.449	1.000

Selbsteinschätzung der Versuchspersonen

Ergänzend zu den im Experiment erzielten Punkten, wird auch analysiert, wie die Versuchspersonen ihre eigenen Programmierkenntnisse einschätzen. Dazu wurden die Selbsteinschätzungsfragen des Fragebogens mittels einer 4-stufigen Likert-Skala beantwortet. Die drei Fragen luateten folgendermaßen:

- *Self1:* Schätzen Sie Ihre allgemeinen Programmierkenntnisse als gut ein?
- *Self2:* Schätzen Sie Ihre Programmierkenntnisse in C als gut ein?
- *Self3:* Schätzen Sie Ihre Fähigkeiten als Reviewer als gut ein?

Die deskriptive Statistik zu den erlangten Ergebnisse findet sich nachfolgend in Tabelle 5.6:

Tabelle 5.6: Selbsteinschätzung der Probanden

Gruppe	Item	Mittelwert	SD	Mdn
Komplett	Self1	2.740	0.860	3.000
	Self2	2.520	0.990	3.000
	Self3	2.430	0.840	3.000
Experten	Self1	3.000	0.930	3.000
	Self2	2.750	1.040	3.000
	Self3	2.880	0.640	3.000
Novizen	Self1	2.600	0.830	3.000
	Self2	2.400	0.990	3.000
	Self3	2.220	0.860	2.000

Bei der Betrachtung der Ergebnisse fällt auf, dass sich die Experten insgesamt als besser, jedoch nicht als überlegen einschätzen. Dies mag darauf zurückgehen, dass C zwar in deren Arbeitsalltag eine Rolle spielt, jedoch in den meisten Fällen nicht so häufig genutzt wird, wie beispielsweise objektorientierte Programmiersprachen (TIOBE, 2022).

5.3.3 Deskriptive Betrachtung der Eye-Tracking-Daten

Die nachfolgenden Absätze beschäftigen sich mit einer globalen Betrachtung der relevanten Eye-Tracking-Metriken, welche auf der Auflistung von Sheridan und Reingold

(2017, S.5) basieren und in Kapitel 4 bei der Bildung von Hypothesen (siehe Absatz 4.2.2) für diese Studie näher präzisiert worden sind. Konkret handelt es sich dabei um die *total* und *average number of fixations*, die *total* und *average duration of fixation*, die *fixation rate*, die *total* und *average number of saccades*, die *total* und *average number of visits on erroneous lines*, sowie die *total* und *average dwell time on erroneous lines*. Die Vorgehensweise folgt dabei einem deskriptiven Ansatz und betrachtet für jede Variable erfahrungsbedingte Unterschiede zwischen den Gruppen der Experten und Novizen.

Deskriptive Statistik zur *total number of fixations*

Tabelle 5.7: Deskriptive Statistik zu total number of fixations

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1525.478	643.452	1418.000	584.000	3197.000	2613.000
Experten	1374.000	718.081	1197.000	584.000	2774.000	2190.000
Novizen	1606.267	610.487	1426.000	914.000	3197.000	2283.000

In Tabelle 5.7 wird die aufgabenübergreifende deskriptive Statistik zur *total number of fixations* dargestellt. Es kann beobachtet werden, dass Experten während des gesamten Experiments insgesamt weniger Fixations benötigen (M= 1374.000; SD=718.081; Mdn=1197.000) als die untersuchten Novizen (M=1606.267; SD=610.487; Med=1426.000).
Bezüglich der Verteilung der *total number of fixations* deutet ein Kolmogorov-Smirnov-Test (D=1.000; p=0.000) darauf hin, dass die Variable nicht normalverteilt ist. Ein ergänzender Saphiro-Wilk-Test liefert widersprüchliche Ergebnisse und legt nahe (W=0.923: p=0.079), dass diese Variable normalverteilt ist. Zur Absicherung wir die in Abbildung 5.1 dargestellte Verteilung betrachtet, welche sich ansatzweise an einer Normalverteilung orientiert, jedoch von dieser zu stark abweicht. Insofern wird im Folgenden die *total number of fixations* als nicht normalverteilt betrachtet.
Mögliche Zusammenhänge zwischen der *total number of fixations*, der angesammelten Erfahrung der Versuchspersonen, sowie der Anzahl der im Experiment erzielten Punkte führen zu keinen statistisch signifikanten Ergebnissen. Für die *total number of fixations* und die *allgemeine* (r_{BP}=-0.127; p=0.563), *professionelle Programmiererfahrung* (r_{BP}=-0.172; p=0.432) ergeben sich lediglich schwache Korrelationskoeffizienten (Cohen, 1988; Cohen, 1992), welche keine Signifikanz erlangen. Selbiges lässt sich hinsichtlich des Korrelationskoeffizienten für die *total number of fixations* und die *Anzahl der erzielten Punkte beobachten* (r_{BP}=-0.027; p=0.903).
Hinsichtlich signifikanter Unterschiede zwischen den beiden Gruppen wurde ein Mann-Whitney-U-Test durchgeführt. Dieser (U=76; z=-0.985; r=0.205; p=0.325) konnte jedoch keine relevanten Unterschiede belegen (Cohen, 1988; Cohen, 1992).

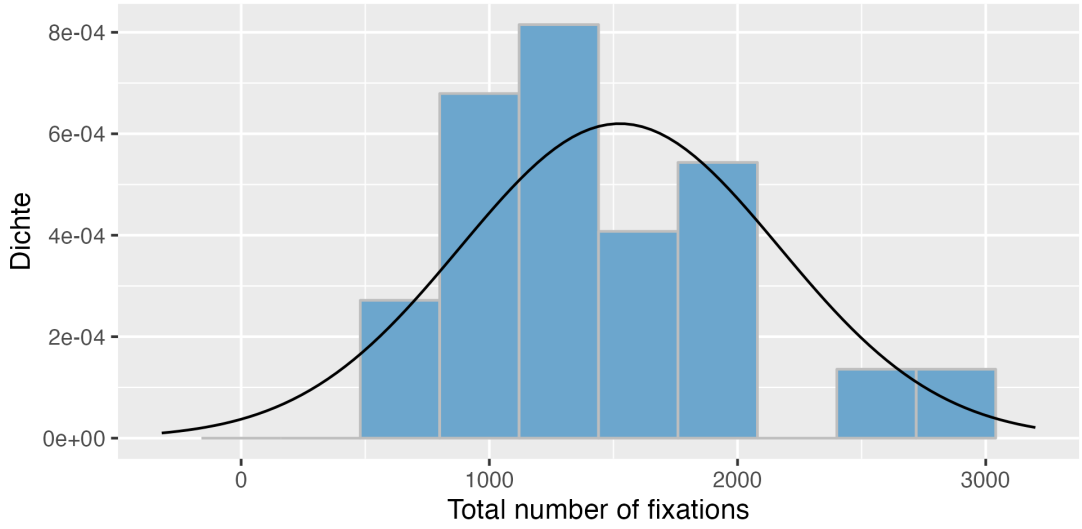

Abbildung 5.1: Verteilung der *total number of fixations*

Tabelle 5.8: Deskriptive Statistik zu average number of fixations

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	84.749	35.747	78.778	32.444	177.611	145.167
Experten	76.333	39.893	66.500	32.444	154.111	121.667
Novizen	89.237	33.916	79.222	50.778	177.611	126.833

Deskriptive Statistik zur *average number of fixations*

Ähnlich zur Betrachtung der *total number of fixations* verhält sich auch die *average number of fixations*. Hier lässt sich erneut beobachten, dass diese für die Experten (M=76.333; SD=39.893; Mdn=66.500) kleiner ausfällt als für Novizen (M=89.237; SD=33.916; Mdn=79.222).
Die Verteilung der *average number of fixations* unterscheidet sich nicht von der zuvor beschriebenen *total number of fixations*. Erneut liegt seitens der Tests ein widersprüchliches Ergebnis vor. Der Kolmogorov-Smirnov-Test (D=1.000; p=0.000) kann keine Normalverteilung nachweisen, der Saphiro-Wilk-Test (W=0.923: p=0.079) hingegen deutet auf eine solche hin. Auch in diesem Fall wird das Ergebnis grafisch überprüft (siehe Abbildung 5.2). Die hier gezeigte Kurve weist eine größere Ähnlichkeit zu einer Normalverteilung auf als die der *total number of fixations* (siehe 5.3), weicht aber ebenfalls von dieser ab. Insofern kann geschlussfolgert werden, dass keine Normalverteilung für die *average number of fixations* vorliegt.
Hinsichtlich möglicher Zusammenhänge wurden erneut Korrelationskoeffizienten berechnet. Es ergibt sich das gleiche Bild, wie bei der *total number of fixations*. Es können lediglich schwache, statistisch nicht signifikante Korrelationskoeffizienten für die *average number of fixations* und die *allgemeine* (r_{BP}=-0.127; p=0.563), sowie für die *professionelle Programmiererfahrung* (r_{BP}=-0.172; p=0.432) erfasst werden. Ein ähnliches Ergebnis ergibt sich für die *Anzahl der erzielten Punkte beobachten* (r_{BP}=-0.027; p=0.903). Ebenso konnten zwischen den beiden Gruppen keine statistisch signifikanten Un-

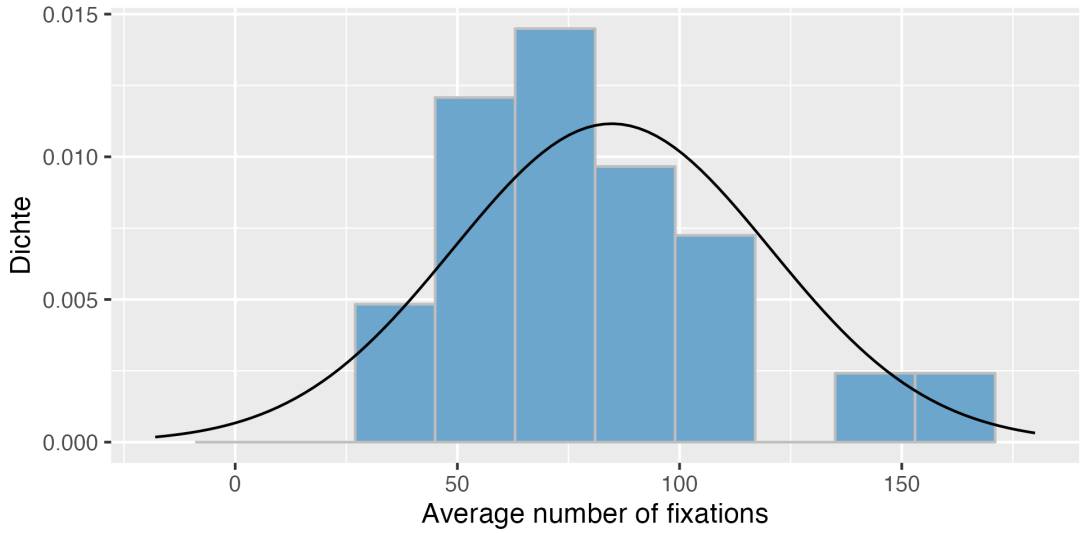

Abbildung 5.2: Verteilung der *average number of fixation*

terschiede beobachtet werden. Ein Mann-Whitney-U-Test (U=76; z=-0.985; r=0.205; p=0.325) lieferte diesbezüglich keine verwertbaren Ergebnisse (Cohen, 1988; Cohen, 1992).

Deskriptive Statistik zur *total fixation duration* in [ms]

Tabelle 5.9: Deskriptive Statistik zu total fixation duration in [ms]

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	573952.244	328438.005	507598.304	149865.629	1466149.613	1316283.984
Experten	570169.135	407404.239	402620.885	149865.629	1293061.787	1143196.158
Novizen	575969.903	294127.092	524149.752	233275.203	1466149.613	1232874.410

Eine Betrachtung der deskriptiven Statistik zur *total fixation duration* erfolgt in Tabelle 5.9. Hier kann beobachtet werden, dass die Mittelwerte beider Gruppen relativ nahe zusammenliegen. Die Novizinnen und Novizen erzielen einen Mittelwert von 575969.903 (SD=294127.092) Millisekunden, wogegen sich dieser für Expertinnen und Experten 570169.135 beläuft. Auffällig ist jedoch, an dieser Stelle, dass die Standardabweichung der Experten mit 407404.239 deutlich höher ausfällt als für die Gruppe der weniger erfahrenen Versuchspersonen. Dies kann auf einzelne statistische Ausreißer in der relativ kleinen Teilstichprobe (n=8) zurückgeführt werden, weshalb die Betrachtung der Mediane im Falle dieser Metrik robustere Ergebnisse gewährleistet. So beträgt der Median für die Expertinnen und Experten 402620.885 Millisekunden und für die Novizinnen und Novizen 524149.752 Millisekunden.
Hinsichtlich der Verteilung kann keine Normalverteilung beobachtet werden (siehe Abbildung 5.3). Sowohl der Kolmogorov-Smirnov-Test (D=1.000; p=0.000), als auch der Saphiro-Wilk-Test (W=0.884; p=0.012) bestätigen dies.
Wie auch bei den vorangegangenen Metriken wurden erneut mögliche Korrelatio-

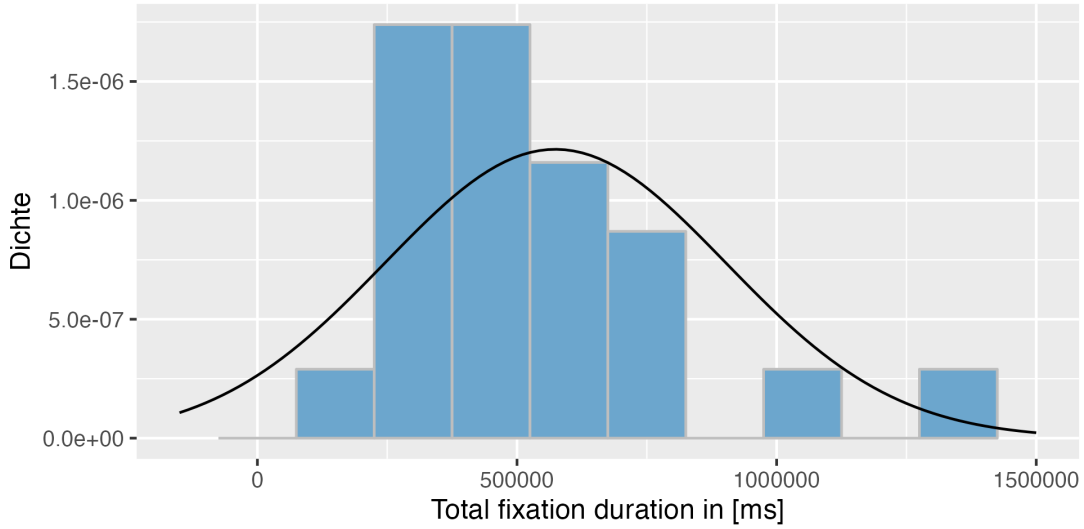

Abbildung 5.3: Verteilung der *total fixation duration*

nen zwischen der *total fixation duration*, sowie der *allgemeinen, professionellen Programmiererfahrung* und der *Anzahl der erzielten Punkte* betrachtet. Es kann lediglich eine schwache und statistisch nicht signifikante Korrelation zwischen der *professionellen Programmiererfahrung* und der *total fixation duration* beobachtet werden (r_{BP}=-0.181; p=0.408). Sowohl für die *Anzahl der erzielten Punkte* (r_{BP}=-0.040; p=0.858), als auch für den Zusammenhang zwischen der *allgemeinen Programmiererfahrung* und der *total fixation duration* (r_{BP}=-0.040; p=0.858) ergeben sich keine relevanten Korrelationskoeffizienten.
Zur Betrachtung möglicher signifikanter Unterschiede zwischen den beiden Gruppen wurde ein Mann-Whitney-U-Test durchgeführt. Dieser kann jedoch keine relevanten Ergebnisse zu Tage fördern (U=69; z=0.537; r=0.112; p=0.591).

Deskriptive Statistik zur *average fixation duration* in [ms]

Tabelle 5.10: Deskriptive Statistik zu average fixation duration in [ms]

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	385.823	138.194	362.961	168.169	729.650	561.481
Experten	388.709	148.695	357.408	230.703	729.650	498.947
Novizen	384.284	137.653	366.496	168.169	717.830	549.662

Hinsichtlich der fixationsbasierten Eye-Tracking-Metriken wurde auch die *average fixation duration* betrachtet (siehe Tabelle 5.10). Aus deskriptiver Sicht liegen hier beide Gruppen ebenfalls relativ eng zusammen und es lassen sich keine Auffälligkeiten zwischen den Experten (M=388.709; SD=148.695; Mdn=357.408) und Novizen (M=384.284; SD=137.653; Mdn=366.496) beobachten.
In Bezug auf die Verteilung zeigt sich erneut, dass diese Variable nicht normal verteilt ist (siehe Abbildung 5.4). Erneut wird dies durch einen Kolmogorov-Smirnov-

(D=1.000; p=0.000) und einen Saphiro-Wilk-Test bestätigt (W=0.897; p=0.021).

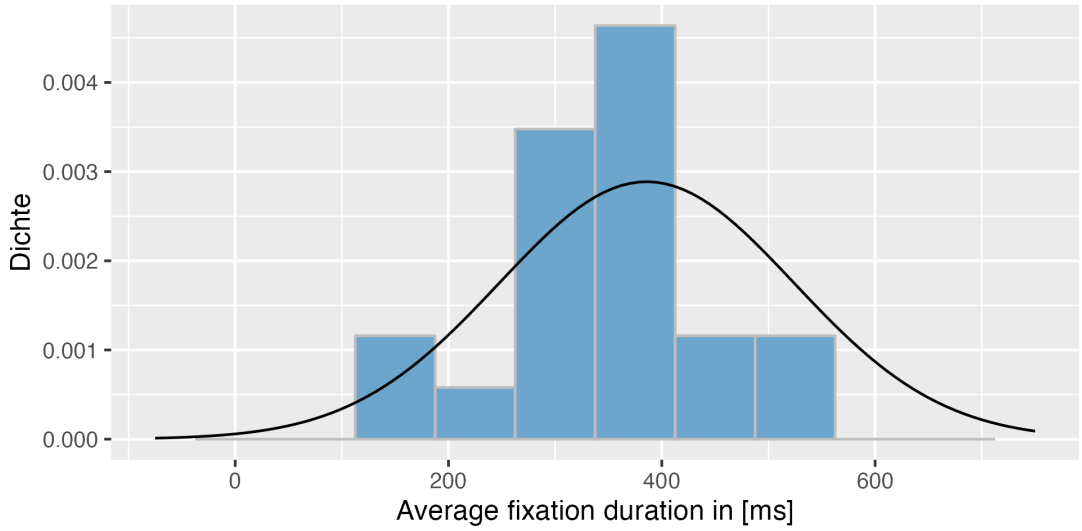

Abbildung 5.4: Verteilung der *average fixation duration*

Wie auch bei den zuvor analysierten Variablen wurde auch im Falle der *average fixation duration* untersucht, ob sich Zusammenhänge mit der angesammelten Erfahrung oder den im Experiment *erzielten Punkten* ergibt. Die Analysen zeigen, dass sich für die *allgemeine Programmiererfahrung* und die *average fixation duration* kein signifikanter Zusammenhang ergibt (r_{BP}=0.014; p=0.951). Die Korrelationskoeffizienten zwischen der *average fixation duration* und der *professionellen Programmiererfahrung* (r_{BP}=-0.186; p=0.395), sowie zwischen der *Anzahl der im Experiment erzielten Punkte* (r_{BP}=-0.153; p=0.485) erlangen zwar keine statistische Signifikanz, deuten aber auf einen schwachen Zusammenhang hin (Cohen, 1988; Cohen, 1992).
Abschließend wurde durch einen Mann-Whitney-U-Test überprüft, ob es zwischen den untersuchten Gruppen signifikante Unterschiede gibt. Diese können durch den Test jedoch nicht nachgewiesen werden (U=65; z=0.284; r=0.059; p=0.776).

Deskriptive Statistik zur *fixation rate*

Tabelle 5.11: Deskriptive Statistik zu fixation rate

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1.387	0.263	1.453	0.689	1.701	1.012
Experten	1.408	0.264	1.467	0.836	1.701	0.865
Novizen	1.377	0.271	1.453	0.689	1.700	1.011

Die letzte analysierte fixationsbasierte Metrik stellt die *fixation rate* dar. Die Betrachtung der deskriptiven Statistik (siehe Tabelle 5.11 zeigt erneut, dass die beiden untersuchten Gruppen relativ ähnliche Werte aufweisen. Die Expertinnen und Experten belaufen sich auf einen Mittelwert von 1.408 *fixations/sec* (SD=0.264; Mdn=1.467), die Novizinnen und Novizen auf 1.377 *fixations/sec* (SD=0.271; Mdn=1.453).

Für die Verteilung der *fixation rate* kann erneut keine Normalverteilung festgestellt werden (siehe Abbildung 5.5). Diese Beobachtung wird erneut durch einen Kolmogorov-Smirnov- (D=0.771; p=0.000) und einen Shapiro-Wilk-Test bekräftigt (W=0.891; p=0.017) bekräftigt.

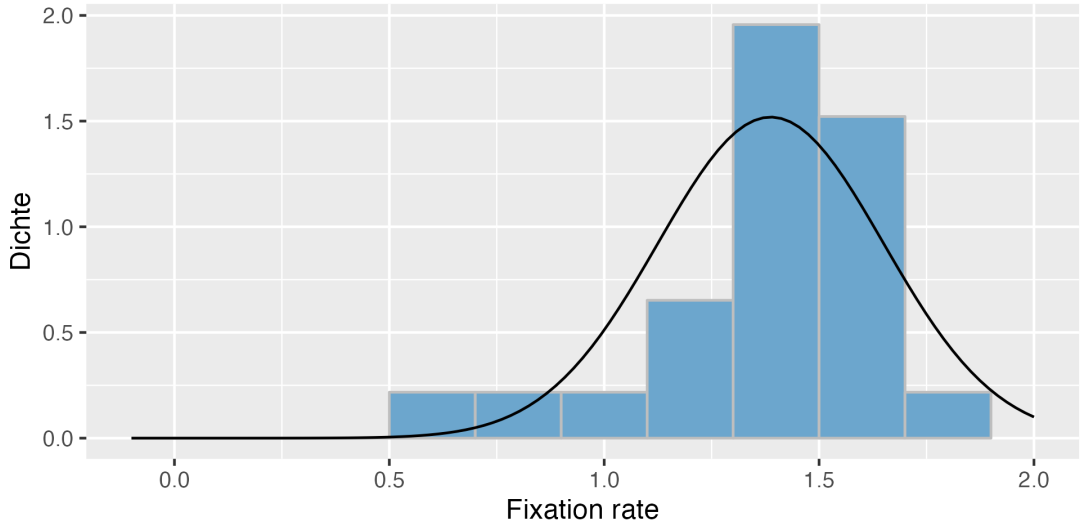

Abbildung 5.5: Verteilung der *fixation rate*

Die Betrachtung der errechneten Korrelationskoeffizienten zeigt schwache (Cohen, 1988; Cohen, 1992), jedoch nicht signifikante Zusammenhänge zwischen der *fixation rate* und der *professionellen Programmiererfahrung* (r_{BP}=0.234; p=0.283), sowie der *Anzahl der erzielten Punkte* r_{BP}=0.217; p=0.320). Für die *fixation rate* und die *allgemeine Programmiererfahrung* kann kein relevanter Zusammenhang festgestellt werden (r_{BP}=0.008; p=0.972).
Mögliche Unterschiede zwischen den beiden unterschiedlichen Erfahrungsstufen wurden erneut über mit einem Mann-Whitney-U-Test untersucht. Dieser kann jedoch keine statistisch signifikanten Unterschiede feststellen (U=57; z=0.158; r=0.033; p=0.875).

Deskriptive Statistik zur *total number of saccades*

Tabelle 5.12: Deskriptive Statistik zu total number of saccades

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1595.087	854.476	1365.000	679.000	4029.000	3350.000
Experten	1271.875	594.217	1100.500	679.000	2545.000	1866.000
Novizen	1767.467	937.693	1419.000	851.000	4029.000	3178.000

Zu Beginn der Betrachtung der saccadenbasierten Metriken wird zuerst die deskriptive Statistik zur *total number of saccades* analysiert (siehe Tabelle 5.12. Hier lassen sich für die Expertinnen und Experten niedrigere Werte (M=1271.875; SD=594.217; Mdn=1100.500) als für die Novizinnen und Novizen (M=1767.467; SD=937.693;

Mdn=1419.000) beobachten. Zusätzlich fällt auf, dass die Gruppe der Novizinnen und Novizen im Vergleich zu den Experten (Max=2545.000) einen deutlich höheren Maximalwert (4029.000) für die *total number of saccades* aufweist.
Die Verteilung der *total number of saccades* ist ebenfalls nicht normalverteilt. Dies wird sowohl grafisch (siehe Abbildung 5.6, als auch durch einen Kolmogorov-Smirnov- (D=1.000; p=0.000) und einen Shapiro-Wilk-Test (W=0.830; p=0.001) bestätigt.

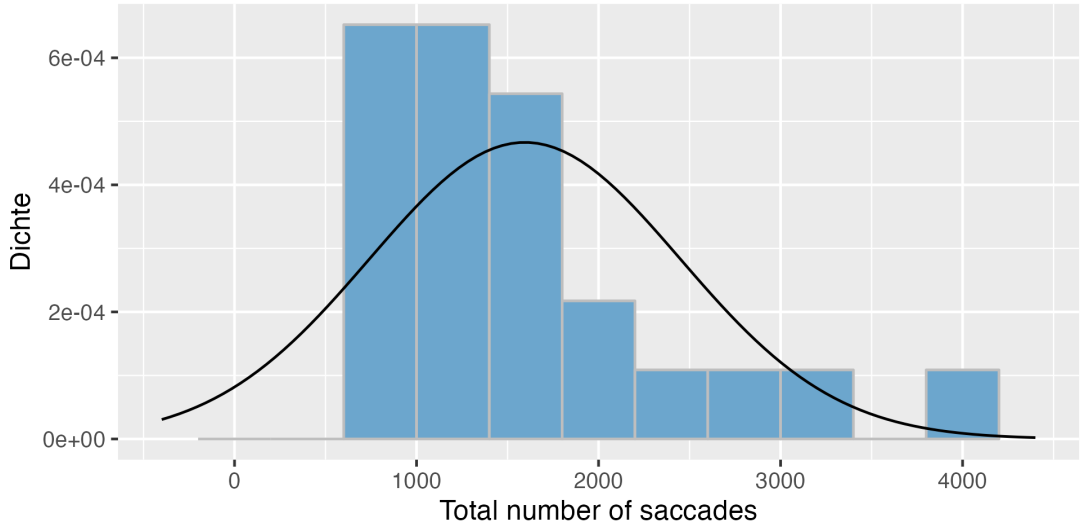

Abbildung 5.6: Verteilung der *total number of saccades*

Hinsichtlich möglicher Korrelationskoeffizienten wurden erneut die Zusammenhänge zwischen der *total number of saccades* und der angesammelten *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der im Experiment erzielten Punkte* betrachtet. Keiner der drei Korrelationskoeffizienten erreicht eine statistische Signifikanz, jedoch lassen sich für die *allgemeine* (r_{BP}=-0.244; p=0.261) und *professionelle Programmiererfahrung* (r_{BP}=-0.267; p=0.217) schwache bis mittlere Effektstärken (Cohen, 1988) beobachten. Die Effektstärke welche sich bezüglich der *Anzahl der erzielten Punkte* ergibt, fällt deutlich mit r_{BP}=0.002 (p=0.990) deutlich schwächer aus.
Mögliche Unterschiede, die sich zwischen den Gruppen ergeben könnten wurden mittels Mann-Whitney-U-Test überprüft (U=83; z=1.442; r=0.301; p=0.149). Es können dabei jedoch keine signifikanten Unterschiede festgestellt werden.

Deskriptive Statistik zur *average number of saccades*

Tabelle 5.13: Deskriptive Statistik zu average number of saccades

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	88.616	47.471	75.833	37.722	223.833	186.111
Experten	70.660	33.012	61.139	37.722	141.389	103.667
Novizen	98.193	52.094	78.833	47.278	223.833	176.556

Hinsichtlich der *average number of saccades* ergibt sich bei der Betrachtung der de-

skriptiven Statistik (siehe Tabelle 5.13 eine ähnliche Situation, wie zuvor bei der *total number of saccades*. Der Mittelwert für die Gruppe der Expertinnen und Experten beläuft sich auf 70.660 (SD=33.012; Mdn=61.139). Die Gruppe der Novizinnen und Novizen erzielt hingegen höhere Werte (M=98.193; SD=52.094; Mdn=78.833).
Die Verteilung der *average number of saccades* gestaltet sich identisch zur *total number of saccades*. Dies wird erneut grafisch (siehe Abbildung 5.7, und durch durch einen Kolmogorov-Smirnov- (D=1.000; p=0.000) und einen Shapiro-Wilk-Test (W=0.830; p=0.001) bestätigt. Es liegt für die *average number of saccades* keine Normalverteilung vor.

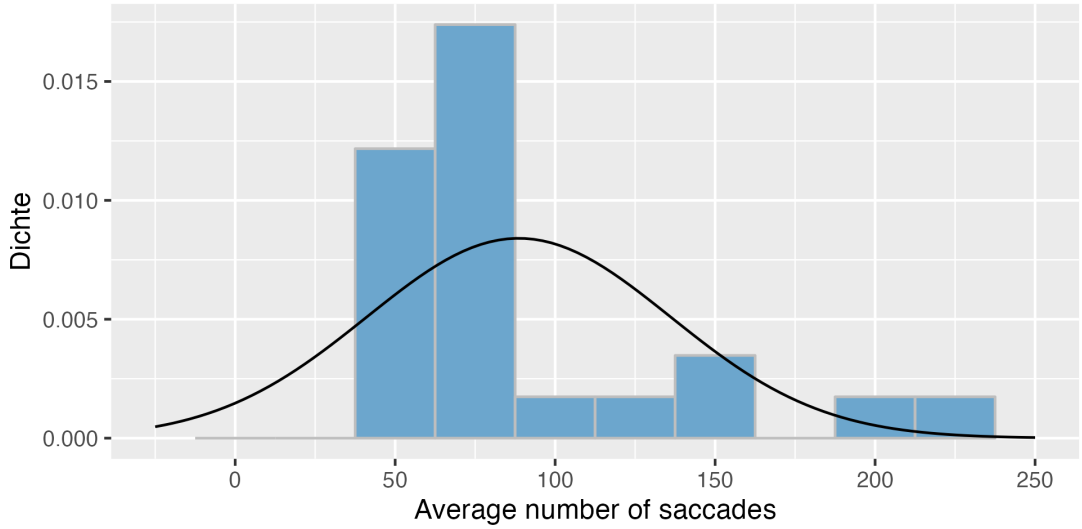

Abbildung 5.7: Verteilung der *average number of saccades*

Bezüglich möglicher Zusammenhänge zwischen der average number of saccades und der *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der im Experiment erzielten Punkte* zeigt sich ebenfalls ein identisches Bild zur zuvor beschriebenen *total number of saccades*: Keiner der drei Korrelationskoeffizienten erreicht eine statistische Signifikanz, es lassen sich jedoch erneut die selben Ergebnisse beobachten. Für die *allgemeine* (r_{BP}=-0.244; p=0.261) und *professionelle Programmiererfahrung* (r_{BP}=-0.267; p=0.217) ergeben sich schwache bis mittlere Effektstärken (Cohen, 1988). Die Effektstärke welche bezüglich der *Anzahl der erzielten Punkte* beobachtet werden kann, fällt mit r_{BP}=0.002 (p=0.990) deutlich schwächer aus.
Zur Überprüfung auf mögliche Unterschiede wurde ein Mann-Whitney-U-Test durchgeführt. Dieser erzielt identische Ergebnisse zur *total number of saccades* (U=83; z=1.442; r=0.301; p=0.149).

Deskriptive Statistik zur *total number of visits on erroneous lines*

Hinsichtlich der Betrachtung der AOI-basierten Variablen wird an dieser Stelle mit der *total number of visits on erroneous lines* begonnen. Die deskriptive Statistik zu dieser (siehe Tabelle 5.14 zeigt leichte Unterschiede zwischen den beiden untersuchten Gruppen auf. Die Expertinnen undExperten erzielen einen Mittelwert von 90.875

Tabelle 5.14: Deskriptive Statistik zu total number of visits on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	104.739	62.212	97.000	22.000	262.000	240.000
Experten	90.875	54.632	71.500	33.000	201.000	168.000
Novizen	112.133	66.493	97.000	22.000	262.000	240.000

(SD=54.632; Mdn=71.500), die Novizinnen und Novizen hingegen einen Mittelwert von 121.133 (SD=66.493; Mdn=97.000).
Auch für die *total number of visits on erroneous lines* zeigt eine Überprüfung der Verteilung, dass keine Normalverteilung vorliegt. Dies wird durch sowohl durch einen Kolmogorov-Smirnov- (D=1.000; p=0.000), einen Shapiro-Wilk-Test (W=0.887; p=0.014) und eine grafische Betrachtung der Verteilungskurve (siehe Abbildung 5.8) bestätigt.

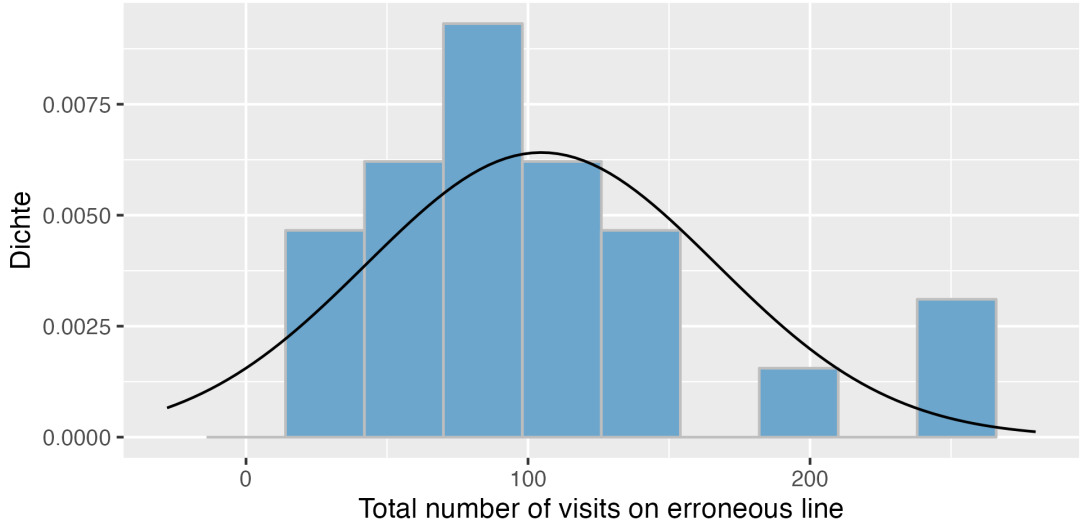

Abbildung 5.8: Verteilung der *total number of visits on erroneous lines*

Die Betrachtung der Korrelationskoeffizienten zeigt schwache, statistisch nicht signifikante Effekte für den Zusammenhang mit der *allgemeinen* (r_{BP}=–0.138; p=0.530) und *professionellen Programmiererfahrung* (r_{BP}=-0.115; p=0.603) auf. für die *Anzahl der im Experimente erzielten Punkte* und die *total number of visits on erroneous lines* ergibt sich kein nennenswerter Zusammenhang (r_{BP}=-0.002; p=0.994)
Die Analyse von Unterschieden zwischen den beiden Gruppen mittels Mann-Whitney-U-Test (U=76; z=1.001; r=0.209; p=0.317) deutet auf keine nennenswerten Unterschiede hin.

Deskriptive Statistik zur *average number of visits on erroneous lines*

Die Betrachtung der deskriptiven Statistik zur *average number of visits on erroneous lines* (siehe Tabelle 5.15 offenbart ein ähnliches Bild zur zuvor beschriebenen *total number of visits on erroneous lines*. Diese fällt für Expertinnen und Experten erneut geringer

Tabelle 5.15: Deskriptive Statistik zu average number of visits on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	5.819	3.456	5.389	1.222	14.556	13.333
Experten	5.049	3.035	3.972	1.833	11.167	9.333
Novizen	6.230	3.694	5.389	1.222	14.556	13.333

aus (M=5.049; SD=3.035; Mdn=3.972), als für Novizinnen und Novizen (M=6.230; SD=3.694; Mdn=5.389).
Ebenso zeigt sich auch für die *average number of visits on erroneous lines* keine Normalverteilung. Der Kolmogorov-Smirnov- (D=0.923; p=0.000) und Shapiro-Wilk-Test (W=0.887; p=0.014) bestätigen dies. Das Ergebnis der Tests wird zusätzlich durch eine grafische Überprüfung abgesichert (siehe Abbildung 5.9).

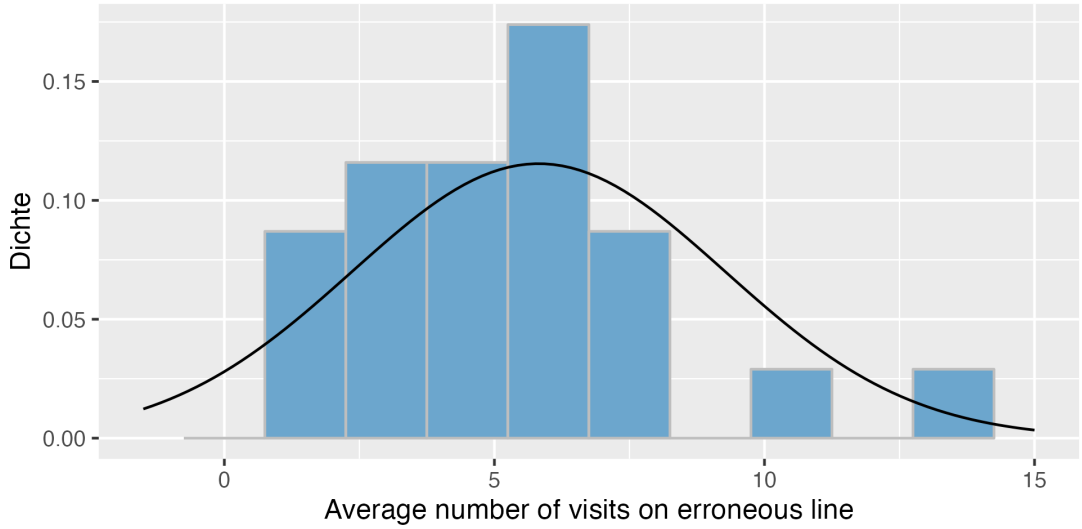

Abbildung 5.9: Verteilung der *average number of visits on erroneous lines*

Die berechneten Korrelationskoeffizienten für die *allgemeine* (r_{BP}=-0.138; p=0.530) und die *professionelle Programmiererfahrung* (r_{BP}=-0.115; p=0.600) bewegen sich nach Cohen (1988) auf einem schwachen Niveau und erreichen keine statistsiche Signifikanz. Schwach fällt auch die Effektstärke des Korrelationskoeffizienten für die *Anzahl der im Experiment erzielten Punkte* aus. Dieser beläuft sich auf r_{BP}=0.002 (p=0.530).
Identisch zur *total number of visits on erroneous lines* verhält sich auch der Mann-Whitney-U-Test, der Unterschiede zwischen den Gruppen hinsichtlich der *average number of visits on erroneous lines* hätte identifizieren sollen. Dieser deutet auf keine nennenswerten Unterschiede hin (U=76; z=1.001; r=0.209; p=0.317).

Deskriptive Statistik zur *total dwell time on erroneous lines* in [ms]

Die Betrachtung der deskriptiven Statistik zur *total dwell time on erroneous lines* in Tabelle 5.16 zeigt, dass die Expertinnen und Experten (M=39903.776; SD=30684.373; Mdn=23952.436) bezüglich dieser Metrik einen niedrigeren Wert aufweisen als die

Tabelle 5.16: Deskriptive Statistik zu total dwell time on erroneous lines in [ms]

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	44603.723	33454.880	34906.884	9994.498	142115.327	132120.829
Experten	39903.776	30684.383	23952.436	10423.253	92069.573	81646.320
Novizen	47110.362	35618.368	39969.983	9994.498	142115.327	132120.829

Gruppe der Novizen (M=47110.362; SD=35618.368; Mdn=39969.983). Auffällig ist an dieser Stelle, dass die Mediane deutlich weiter nach unten korrigiert wurden, was selbst innerhalb der Gruppen auf relativ starke Schwankungen in der Betrachtungsdauer der fehlerhaften Zeile schließen lässt.
Erneut kann keine Normalverteilung für diese Metrik festgestellt werden. Dies kann durch einen Kolmogorov-Smirnov- (D=1.000; p=0.000), einen Shapiro-Wilk-Test (W=0.859) und eine grafische Betrachtung der Verteilung (siehe Abbildung 5.10) belegt werden.

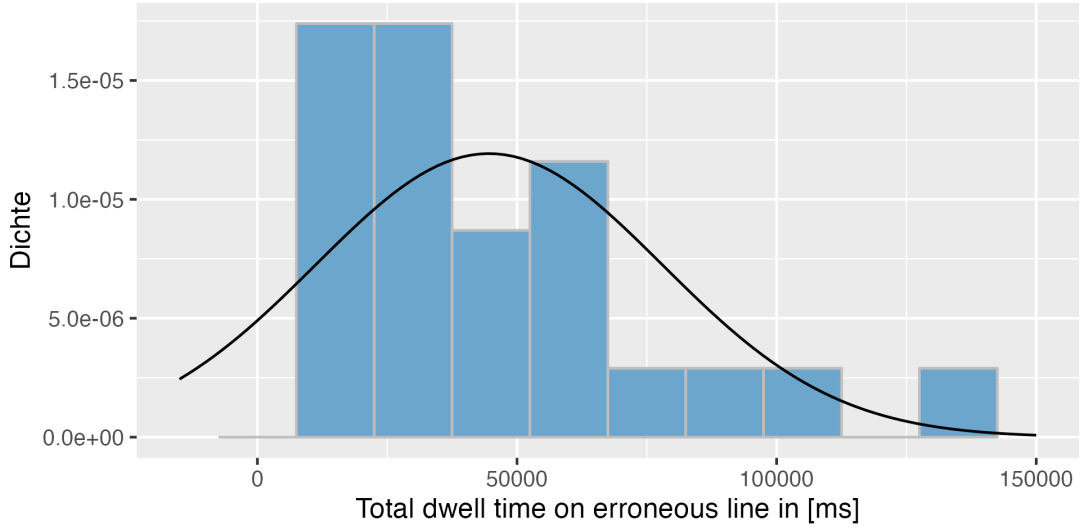

Abbildung 5.10: Verteilung der *total dwell time on erroneous lines*

Die Überprüfung der Korrelationskoeffizienten zwischen der *total dwell time on erroneous lines* und der angesammelten *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der im Experiment erzielten Punkte* bringt keine statistisch signifikanten Ergebnisse hervor. Nennenswert ist an dieser Stelle die schwache, jedoch nicht signifikante Korrelation zwischen der *professionellen Programmiererfahrung* und der *total dwell time on erroneous lines* (r_{BP}=-0.187; p=0.394). Für die *allgemeine Programmiererfahrung* r_{BP}=-0.044; p=0.843) und die *Anzahl der erzielten Punkte* r_{BP}=0.050; p=0.819) treten keine relevanten Korrelationen zur *total dwell time on erroneous lines* auf.
Die Überprüfung auf statistisch signifikante Unterschiede zwischen den Gruppen mittels eines Mann-Whitney-U-Tests deutet auf keine relevanten Unterschiede hin (U=68; z=0.474; r=0.100; p=0.636).

Deskriptive Statistik zur *average dwell time on erroneous lines*

Tabelle 5.17: Deskriptive Statistik zu average dwell time on erroneous lines in [ms]

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	2477.985	1858.604	1939.271	555.250	7895.296	7340.046
Experten	2216.876	1704.688	1330.691	579.070	5114.976	4535.907
Novizen	2617.242	1978.798	2220.555	555.250	7895.296	7340.046

Die deskriptive Statistik zu den grundlegenden Eye-Tracking-Metriken endet mit der *average dwell time on erroneous lines*. Diese findet sich in Tabelle 5.17. Bei der Betrachtung dieser Tabelle zeigt sich erneut, dass die Gruppe der Expertinnen und Experten im Durchschnitt weniger Zeit für die Betrachtung der fehlerhaften Zeilen aufwendet (M=2261.876; SD=1704.688; Mdn=1330.691) als die Novizinnen und Novizen (M=2617.242; SD=1978.798; Mdn=2220.555).
Eine Normalverteilung kann für die *average dwell time on erroneous lines* ebenfalls nicht nachgewiesen werden. Dies zeigt sich durch einen Kolmogorov-Smirnov- (D=1.000; p=0.000) und einen Shapiro-Wilk-Test (W=0.858; p=0.004). Eine weitere Bestätigung findet sich in der grafischen Überprüfung der Verteilung (siehe Abbildung 5.11), der ebenfalls keine Normalverteilung zugrunde liegt.

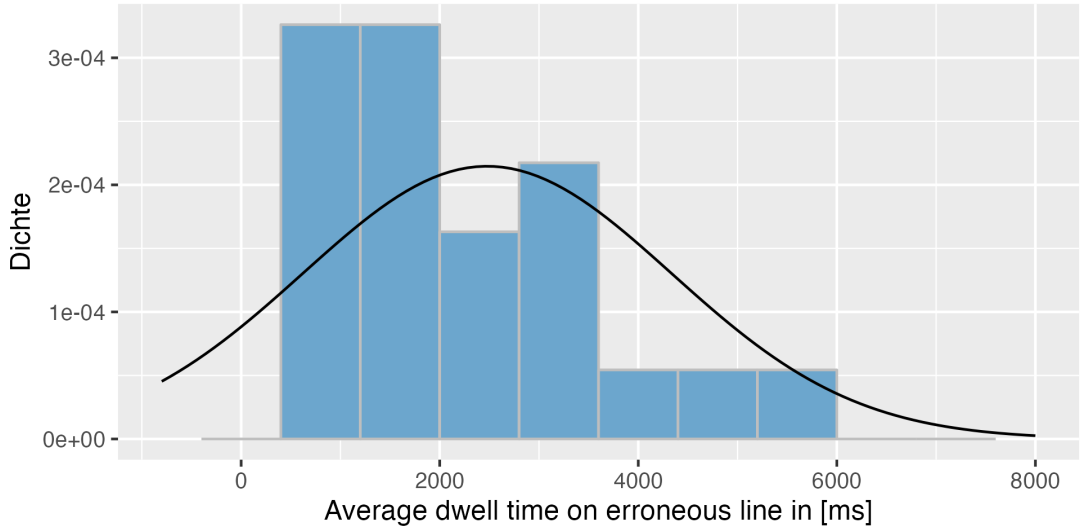

Abbildung 5.11: Verteilung der *average dwell time on erroneous lines*

Hinsichtlich möglicher Korrelationskoeffizienten wurden erneut die Zusammenhänge zwischen der *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der erzielten Punkte* mit der *average dwell time on erroneous lines* untersucht. Diese verhalten sich identisch zur zuvor beschriebenen *total dwell time on erroneous lines*. Es zeigt sich zwischen der *professionellen Programmiererfahrung* und der *average dwell time* erneut eine schwache, nicht signifikante Korrelation (r_{BP}=-0.187; p=0.394). Für die *allgemeine Programmiererfahrung* r_{BP}=-0.044; p=0.843) und die *Anzahl der erzielten Punkte* r_{BP}=0.050; p=0.819) treten auch für die *total dwell time on erroneous lines* keine

relevanten Zusammenhänge auf.
Der zur Überprüfung von statistisch signifikanten Unterschieden zwischen den Gruppen durchgeführte Mann-Whitney-U-Tests deutet auch bei der *average dwell time on erroneous lines* auf keine relevanten Unterschiede hin (U=68; z=0.474; r=0.100; p=0.636).

5.3.4 Phasenbasierte Betrachtung der Eye-Tracking-Daten

Im Anschluss an die im vorherigen Absatz geschilderte Betrachtung der deskriptiven Statistik der Eye-Tracking-Metriken (siehe Absatz 5.3.3) werden nachfolgend die einzelnen Phasen der Code Reviews analysiert.
Die Idee einer phasenbasierten Betrachtung geht zurück auf die Arbeit von Uwano et al. (2006), welche sich vorrangig der Bedeutung des *scan pattern* widmete. Im Falle der vorliegenden Dissertation liegt der Fokus der Analyse jedoch darauf, wie sich die Augenbewegungen der Versuchspersonen bzw. die damit assoziierten Eye-Tracking-Metriken im Verlauf des Code Reviews verändern. Diese Vorgehensweise birgt jedoch eine Schwierigkeit: Auf welcher Basis lassen sich die Variablen in Phasen einteilen?
Aufgrund der starken Streuungen der Eye-Tracking-Daten (siehe Absatz 5.3.3) und mangelnder Vergleichbarkeit mit anderen Domänen (Gegenfurtner et al., 2011; Sheridan & Reingold, 2017) bat es sich nur bedingt an, die Einteilung der Phasen an diesen festzumachen. Um den Grundgedanken der im Unterkapitel 2.3 beschriebenen *holistic models of image perception*(Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000; Swensson, 1980; Sheridan & Reingold, 2017) wahren zu können und Gruppenvergleiche zwischen den Experten und Novizen zu ermöglichen, wurde entschieden, dass die Einteilung in Phasen auf Basis der *individuellen Bearbeitungszeit* erfolgen soll. Dieser Wert streut zwar ebenfalls, ermöglicht aber die bestmögliche zeitliche Betrachtung der Veränderungen der auftretenden Augenbewegungen.
Weiterhin galt es abzuwiegen, wie viele Phasen notwendig sind um eventuelle Veränderungen beobachten zu können. Aus Sicht der Originalstudie von Uwano et al. (2006), sowie auf Grundlage der *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000; Swensson, 1980; Sheridan & Reingold, 2017) kommt vor allem der Anfangsphase eine große Bedeutung zu. Diese ist meist durch einen *Scan* geprägt, durch den sich die Versuchspersonen einen Überblick über den zu begutachtenden Stimulus verschaffen und welcher sich in den Augenbewegungen bemerkbar machen sollte. Ebenso stellt die Suche nach Fehlern (*error detection*) einen weiteren Aspekt dar, welcher abgedeckt werden soll. Abschließend muss auch bedacht werden, dass es im Software Engineering und in der Programmierung üblich ist, dass der eigene Code von fremden Entwicklern im Rahmen eines Reviews kontrolliert wird. Um mögliche Fehler bereits frühzeitig abzufangen, wird Code im Regelfall erst vom verantwortlichen Programmierer oder der Programmiererin überprüft (*verification*), bevor die Freigabe für eine erweiterte Begutachtung erfolgt (Carullo, 2020; Czerwonka et al., 2015; Indriasari et al., 2020; Sommerville, 2012; Tufano et al., 2021). Daraus resultieren drei Tätigkeiten, die sich in den Augen-

bewegungen der Versuchspersonen bemerkbar machen (Hauser, Reuter et al., 2018):

- *Scan:* Die Versuchsperson verschafft sich einen Überblick über den zu begutachtenden Stimulus. Diese Tätigkeit wird mit dem Beginn der Betrachtung assoziiert.
- *Error detection:* Die Versuchsperson sucht im Stimulus nach Auffälligkeiten, Anomalien oder Fehlern. Diese Tätigkeit setzt ein gewisses Grundverständnis des Stimulus voraus.
- *Verification:* Die Versuchsperson überprüft die Resultate der Begutachtung. Dabei wird geklärt, ob die entdeckten Auffälligkeiten Bezug zu anderen Bereichen des Stimulus haben oder ob in diesem noch weitere Anomalien vorhanden sind.

Da die Unterteilung der Rohdaten des Eye-Trackers über das eigens entworfene R-Skript mit einem verhältnismäßig hohem Programmieraufwand verbunden ist, in dessen Rahmen auch spezielle Filterungen durchgeführt werden müssen, spielen hinsichtlich der Entscheidung auch testökonomische Aspekte eine Rolle. Es galt abzuwiegen, ob mit drei oder fünf Phasen gearbeitet werden soll. Die Entscheidung fiel letztendlich auf drei Phasen, da diese eventuelle Tendenzen bereits identifizieren können und die Erstellung eines passenden R-Skripts für eine Person noch zu bewältigen ist. Dennoch sollte für zukünftige Arbeiten eine Unterteilung in fünf Phasen oder die Einbeziehung von KI-Algorithmen (siehe Absatz 8.3.2) in Betracht gezogen werden.
Hinsichtlich der genannten Phasen gilt es ebenfalls die Arbeit von Begel und Vrzakova (2018) zu beachten. Diese geht davon aus, dass sich die verschiedenen Tätigkeiten und die damit verbundenen Augenbewegungen im Verlauf eines Reviews wiederholen.
Nachfolgend soll dargestellt werden, ob sich im Falle der relevanten Eye-Tracking-Metriken statistisch signifikante Unterschiede zwischen den Phasen ergeben. Diese Überprüfung basiert auf einer Reihe von Friedman-Tests, welche durch ergänzende deskriptive Statistik zur besseren Einordnung der Ergebnisse ergänzt wird. Im Rahmen dieser Auswertung wird ebenfalls überprüft, ob sich zwischen den unterschiedlichen Erfahrungsgruppen signifikante Unterschiede ergeben.

Phasenbasierte Betrachtung der *total number of fixations*

Wie bereits in der zuvor beschriebenen deskriptiven Analyse der der Eye-Tracking-Daten (siehe Absatz 5.3.3) wird auch im Falle der phasenbasierten Betrachtung erneut mit der *total number of fixations* begonnen. Die in Tabelle 5.18 dargestellten Friedman-Tests können für diese Metrik keine signifikanten Unterschiede nachweisen.
Dieses Ergebnis verdeutlicht sich durch einen Blick auf die begleitende deskriptive Statistik (siehe Tabelle 5.19 und Abbildung 5.12). Die *total number of fixations* liegt übergreifend über die betrachteten Phasen bei allen Gruppen sehr nahe zusammen. In keiner der Phasen zeigen sich relevante Auffälligkeiten. Ähnlich zur allgemeinen

Tabelle 5.18: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of fixations

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.730)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.802)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
	$FT_{(Gesamt)}$: $\chi^2(2)$=2.835; p=0.243 $FT_{(Experten)}$: $\chi^2(2)$=2.250; p=0.325 $FT_{(Novizen)}$: $\chi^2(2)$=1.322; p=0.516		

Betrachtung der *total number of fixations* zeigt sich auch in den Phasen, dass die Expertinnen und Experten die Aufgaben insgesamt mit weniger Fixations lösen können.

Tabelle 5.19: Deskriptive Statistik zur phasenbasierten Betrachtung der total number of fixations

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	507.435	213.047	491.000	197.000	1028.000	831.000
	2	500.348	208.191	455.000	169.000	1025.000	856.000
	3	517.696	226.120	474.000	218.000	1144.000	926.000
Experten	1	457.375	242.389	399.000	197.000	944.000	747.000
	2	448.250	232.109	394.500	169.000	887.000	718.000
	3	468.375	245.085	403.500	218.000	943.000	725.000
Novizen	1	534.133	199.383	493.000	302.000	1028.000	726.000
	2	528.133	196.966	461.000	296.000	1025.000	729.000
	3	544.000	219.506	482.000	316.000	1144.000	828.000

Phasenbasierte Betrachtung der *average number of fixations*

Vergleichbar zur phasenbasierten Betrachtung der *total number of fixations* verhält sich auch die *average number of fixations*. Die Friedman-Tests weisen für diese Metrik keine statistisch signifikanten Unterschiede zwischen den einzelnen Phasen nach.
Auch im Falle der *average number of fixations* liegen die Werte relativ nahe zusammen. Es zweigen sich weder zwischen den Experten und Novizen, noch in den einzelnen Phasen gravierende Unterschiede. Lediglich die effizientere Performance der Expertinnen und Experten zeichnet sich erneut ab. Die begleitende deskriptive Statistik wird in Tabelle 5.21 ausführlich dargestellt und in Abbildung 5.13 in Form eines Boxplots illustriert.

Phasenbasierte Betrachtung der *total fixation duration in [ms]*

Für die phasenbasierte Betrachtung der *total fixation duration in [ms]* ergeben sich hinsichtlich der untersuchten Phasen und Gruppen mehrere Unterschiede. Diese werden

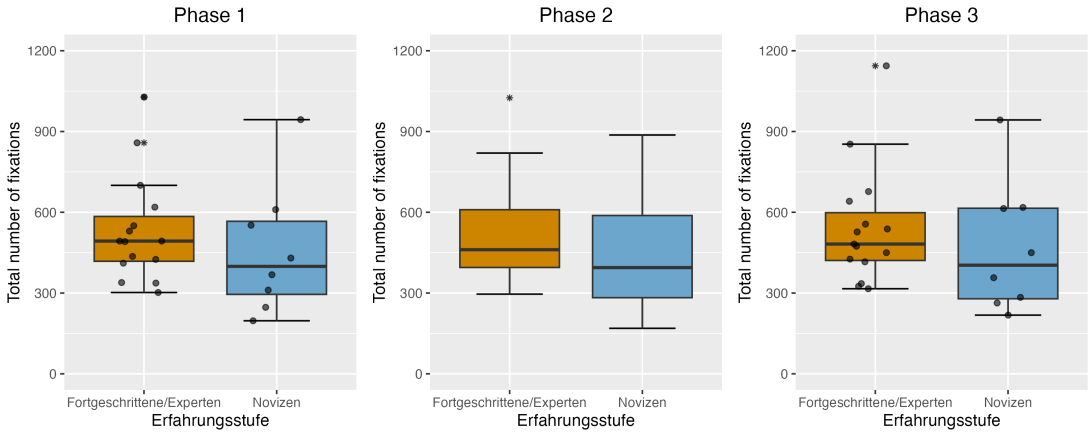

Abbildung 5.12: Betrachtung der phasenbasierten Daten zur *total number of fixations*

Tabelle 5.20: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average number of fixations

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.730)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.802)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=2.835; p=0.243 $FT_{(Experten)}$: $\chi^2(2)$=2.250; p=0.325 $FT_{(Novizen)}$: $\chi^2(2)$=1.322; p=0.516			

durch eine Reihe von Friedman-Tests belegt (siehe Tabelle 5.22).
Übergreifend über die beiden Erfahrungsstufen zeigt sich durch den Friedman-Test ($FT_{(Gesamt)}$: $\chi^2(2)$=23.304; p=0.000) mit begleitenden Post-hoc-Tests, dass sich sich Phase 1 und 2 (p=0.000), sowie Phase 1 und 3 (p=0.000) signifikant voneinander unterscheiden. Dies lässt sich auch in der begleitenden deskriptiven Statistik beobachten (siehe Tabelle 5.23: Phase 1 weist mit 196067.500 (SD=109079.900; Mdn=172566.100) Millisekunden einen deutlich höheren Mittelwert für die *total fixation duration* auf, als Phase 2 (M=190250.800; SD=110594.100; Mdn=168975.600) und 3 (M=187633.900; SD=109099.900; Mdn=165037.800).
Für die Gruppe der Expertinnen und Experten lässt sich durch den Friedman-Test ($FT_{(Experten)}$: $\chi^2(2)$=10.750; p=0.005) lediglich ein signifikanter Unterschied feststellen. In diesem Fall unterscheidet sich Phase 1 signifikant von Phase 3 (p=0.007). Hinsichtlich der deskriptiven Statistik zeigt sich, dass Phase 1 mit einem Mittelwert von 193738.700 (SD=134999.500; Mdn=140756.300) Millisekunden länger ausfällt als Phase 2 (M=190062.600; SD=137565.600; Mdn=130923.800) und 3 (M=186367.800; SD=134945.100; Mdn=130940.800).
Bei der Gruppe der Novizinnen und Novizen können zwei Unterschiede beobachtet werden ($FT_{(Novizen)}$: $\chi^2(2)$=12.933; p=0.002), wobei jedoch lediglich der Unterschied von Phase 1 zu Phase 3 statistische Signifikanz (p=0.003) erlangt.

Tabelle 5.21: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of fixations

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	84.572	35.508	81.833	32.833	171.333	138.500
	2	83.391	34.698	75.833	28.167	170.833	142.667
	3	86.283	37.687	79.000	36.333	190.667	154.333
Experten	1	76.229	40.398	66.500	32.833	157.333	124.500
	2	74.708	38.685	65.750	28.167	147.833	119.667
	3	78.062	40.847	67.250	36.333	157.167	120.833
Novizen	1	89.022	33.230	82.167	50.333	171.333	121.000
	2	88.022	32.828	76.833	49.333	170.833	121.500
	3	90.667	36.584	80.333	52.667	190.667	138.000

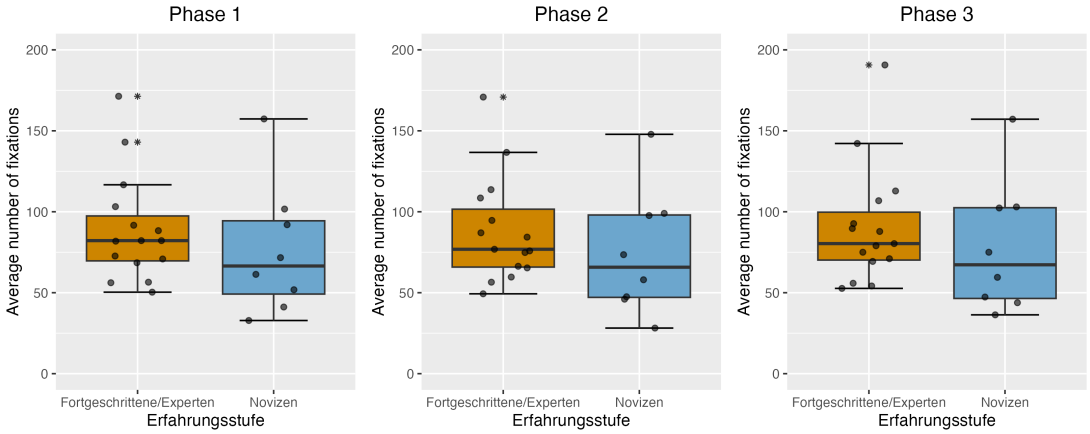

Abbildung 5.13: Betrachtung der phasenbasierten Daten zur *average number of fixations*

Obwohl der Post-hoc-Test einen Unterschied zwischen Phase 1 und 2 anzeigt, kann für diesen keine Signifikanz (p=0.064) nachgewiesen werden. Hinsichtlich der deskriptiven Statistik kann auf Seite der Novizen beobachtet werden, dass auch in ihrem Fall die *total fixation duration in [ms]* für die erste Phase (M=197309.600; SD=97879.240; Mdn=173846.600) deutlich länger ausfällt als für die zweite (M=190351.200; SD=98782.820; Mdn=169994.400) und dritte Phase (188309.200; 97968.780).
Die deskriptive Statistik wird auch für die phasenbasierte Betrachtung der *total fixation duration* erneut durch einen Boxplot dargestellt (siehe Abbildung 5.14).

Phasenbasierte Betrachtung der *average fixation duration in [ms]*

Auf die *total fixation duration* folgend wird die *average fixation duration* betrachtet. Die Friedman-Tests zeigen erneut Unterschiede zwischen den einzelnen Phasen der Gruppen an (siehe Tabelle 5.24).
Gruppenübergreifend betrachtet (siehe Tabelle 5.25) zeigt der Friedman-Test in Kombination mit einem Post-hoc-Test ($FT_{(Gesamt)}$: $\chi^2(2)=9.652$; p=0.008) einen signifikanten Unterschied (p=0.012) zwischen der ersten (M=395.589; SD=133.443; Mdn= 371.134)

Tabelle 5.22: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total fixation duration

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.000)	Unterschied (p=0.000)	Kein Unterschied (p=0.842)
Experten	Kein Unterschied (p=0.273)	Unterschied (p=0.007)	Kein Unterschied (p=1.000)
Novizen	Unterschied (p=0.064)	Unterschied (p=0.003)	Kein Unterschied (p=1.000)

$FT_{(Gesamt)}$: $\chi^2(2)$=23.304; p=0.000
$FT_{(Experten)}$: $\chi^2(2)$=10.750; p=0.005
$FT_{(Novizen)}$: $\chi^2(2)$=12.933; p=0.002

Tabelle 5.23: Deskriptive Statistik zur phasenbasierten Betrachtung der total fixation duration in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	196067.500	109079.900	172566.100	51824.120	496644.400	444820.300
	2	190250.800	110594.100	168975.600	47189.210	487227.300	440038.100
	3	187633.900	109099.900	165037.800	50852.290	482277.800	431425.600
Experten	1	193738.700	134999.500	140756.300	51824.120	433427.500	381603.400
	2	190062.600	137565.600	130923.800	47189.210	429956.100	382766.900
	3	186367.800	134945.100	130940.800	50852.290	429678.200	378825.900
Novizen	1	197309.600	97879.240	173846.600	84236.650	496644.400	412407.800
	2	190351.200	98782.820	169994.400	59903.560	487227.300	427323.800
	3	188309.200	97968.780	183559.800	69151.590	482277.800	413126.300

und dritten Phase (M=378.644; SD=157.061; Mdn=358.100) an.
Für die Expertinnen und Experten ergibt sich im Falle der phasenbasierten Betrachtung *average fixation duration* ($FT_{(Experten)}$: $\chi^2(2)$=7.750; p=0.021) ein signifikanter Unterschied (p=0.036) zwischen der zweiten (M=397.315; SD=161.577; Mdn=361.774) und dritten Phase (M=374.078; SD=148.574; Mdn=340.892).
Bezüglich der Novizinnen und Novizen wird seitens des Post-hoc Tests erneut ein Unterschied festgestellt. Dieser befindet sich zwischen Phase 1 (M=89.022; SD=33.230; Mdn=82.167) und Phase 3 (M=90.667; SD=36.584; Mdn=80.333), erlangt aber keine statistische Signifikanz (p=0.064).
Bei der Betrachtung deskriptiven Statistik fällt auf, dass die phasenbasierte *average fixation duration in [ms]* für die Gruppe der Novizinnen und Novizen deutlich kürzer ausfällt, als für die erfahreneren Versuchspersonen. Eine Reihe von Mann-Whitney-U-Tests kann jedoch keine signifikanten Unterschiede zwischen den beiden Gruppen bestätigen (Phase 1: U=64; p=0.825; Phase 2: U=63; p=0.875; Phase 3: W=69; p=0.591).
Ebenso in der Beschreibung der zuvor genannten Metriken, werden die Ergebnisse zur phasenbasierten Betrachtung der *average fixation duration* grafisch dargestellt. Ein entsprechender Boxplot findet sich in der Abbildung 5.15.

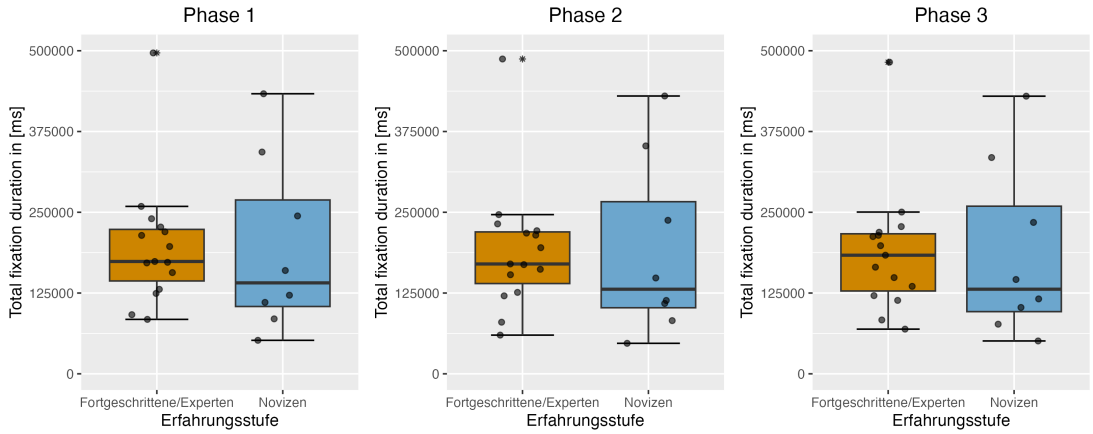

Abbildung 5.14: Betrachtung der phasenbasierten Daten zur *total fixation duration*

Tabelle 5.24: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average fixation duration

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Unterschied (p=0.012)	Kein Unterschied (p=0.461)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.481)	Unterschied (p=0.036)
Novizen	Kein Unterschied (p=0.106)	Unterschied (p=0.064)	Kein Unterschied (p=1.000)
	$FT_{(Gesamt)}$: $\chi^2(2)$=9.652; p=0.008		
	$FT_{(Experten)}$: $\chi^2(2)$=7.750; p=0.021		
	$FT_{(Novizen)}$: $\chi^2(2)$=8.133; p=0.002		

Phasenbasierte Betrachtung der *fixation rate*

Im Zuge der phasenbasierten Betrachtung stellt die *fixation rate* die letzte (vorwiegend) fixationsbasierte Eye-Tracking-Metrik dar. Auch hier wird erneut eine Reihe von Friedman-Tests durchgeführt, welche mehrere Unterschiede hinsichtlich der untersuchten Phasen anzeigen.

Für die gruppenübergreifende Überprüfung mittels Friedman-Test mit anschließendem Post-hoc-Test ($FT_{(Gesamt)}$: $\chi^2(2)$=46.000; p=0.000) ergeben sich Unterschiede zwischen allen drei Phasen. Phase 1 (M=2.283; SD=0.444; Mdn=2.451) unterscheidet sich sowohl von von Phase 2 (M=1.121; SD=0.239; Mdn=1.155) (p=0.004), als auch von Phase 3 (M=0.758; SD=0.132; Mdn=0.792) (p=0.000) signifikant. Auch zwischen Phase 2 und 3 lässt sich ein signifikanter Unterschied beobachten (p=0.000).

Für die Gruppe der Expertinnen und Experten lässt sich nur ein statistisch signifikanter Unterschied bei der *fixation rate* beobachten ($FT_{(Experten)}$: $\chi^2(2)$=16.000; p=0.000). Dieser Unterschied (p=0.000) findet sich zwischen der ersten (M=2.327; SD=0.432; Mdn=2.503) und dritten Phase (M=0.777; SD=0.146; Mdn=0.810).

Der für die Novizinnen und Novizen durchgeführte Friedman-Test ($FT_{(Novizen)}$: $\chi^2(2)$=30.000; p=0.000) zeigt Unterschiede zwischen allen drei Phasen auf. Phase 1 (M=2.260; SD=0.432; Mdn=2.503) unterscheidet sich signifikant von Phase

Tabelle 5.25: Deskriptive Statistik zur phasenbasierten Betrachtung der average fixation duration in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	395.589	133.443	371.134	172.045	699.647	527.602
	2	383.236	137.039	349.082	154.278	769.228	614.951
	3	378.644	157.061	358.100	167.570	900.431	732.861
Experten	1	394.733	139.470	355.530	229.001	699.647	470.646
	2	397.315	161.577	361.774	238.331	769.228	530.898
	3	374.078	148.574	340.892	224.777	720.075	495.297
Novizen	1	89.022	33.230	82.167	50.333	171.333	121.000
	2	88.022	32.828	76.833	49.333	170.833	121.500
	3	90.667	36.584	80.333	52.667	190.667	138.000

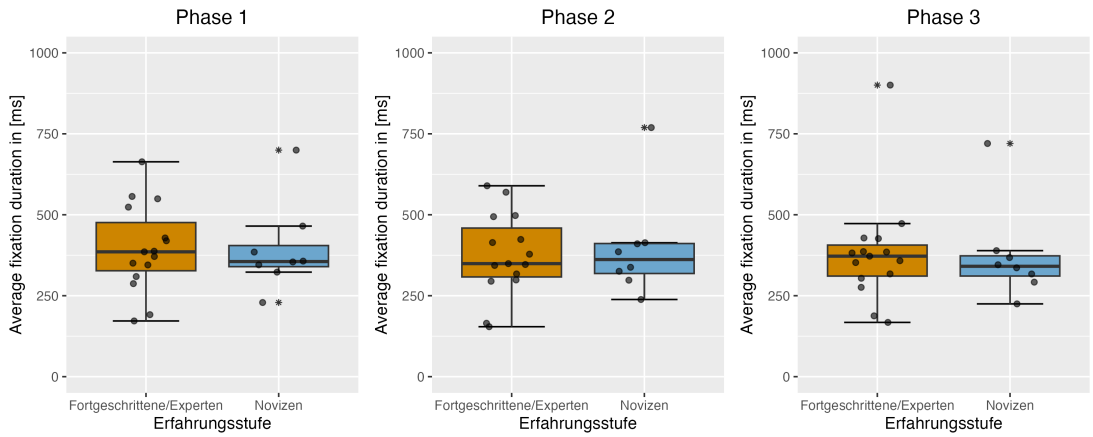

Abbildung 5.15: Betrachtung der phasenbasierten Daten zur *average fixation duration*

2 (M=1.121; SD=0.245; Mdn=1.192) (p=0.004) und Phase 3 (M=0.748; SD=0.128; Mdn=0.755) (p=0.000). Auch der Unterschied zwischen Phase 2 und 3 befindet sich auf einem statistisch signifikanten Niveau (p=0.000).
Eine nähere Darstellung der deskriptiven Statistik kann der Tabelle 5.27 entnommen werden. Diese Tabelle zeugt auch, auf, dass die *fixation rate* im Verlauf der Code Reviews abnimmt und sich unabhängig vom Erfahrungsgrad der Versuchspersonen äußerst ähnlich verhält.
Das Verhalten der *fixation rate* wird auch grafisch dargestellt und verdeutlicht den in Tabelle 5.27 beschriebenen Trend. Dies kann der Abbildung 5.16 entnommen werden.

Phasenbasierte Betrachtung der *total number of saccades*

Die phasenbasierte Betrachtung der saccadenbasierten Metriken beginnt mit der *total number of saccades*. Für diese konnten mittels der Friedman-Tests keine signifikanten Unterschiede festgestellt werden (siehe Tabelle 5.28).
Obwohl sich in den Phasen keine Unterschiede bemerkbar machen, kann in der deskriptiven Statistik (siehe Tabelle 5.29 beobachtet werden, dass die erfahreneren Versuchspersonen mit weniger Saccaden in allen Phasen das Experiment abschließen.

Tabelle 5.26: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der fixation rate

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.004)	Unterschied (p=0.000)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=0.273)	Unterschied (p=0.000)	Kein Unterschied (p=0.273)
Novizen	Unterschied (p=0.004)	Unterschied (p=0.000)	Unterschied (p=0.004)
	$FT_{(Gesamt)}$: $\chi^2(2)$=46.000; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=16.000; p=0.000 $FT_{(Novizen)}$: $\chi^2(2)$=30.000; p=0.000		

Tabelle 5.27: Deskriptive Statistik zur phasenbasierten Betrachtung der fixation rate

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	2.283	0.444	2.451	1.116	2.875	1.759
	2	1.121	0.239	1.155	0.594	1.500	0.906
	3	0.758	0.132	0.792	0.356	0.926	0.569
Experten	1	2.327	0.432	2.503	1.418	2.682	1.264
	2	1.120	0.244	1.124	0.630	1.496	0.866
	3	0.777	0.146	0.810	0.460	0.926	0.466
Novizen	1	2.260	0.464	2.439	1.116	2.875	1.759
	2	1.121	0.245	1.192	0.594	1.500	0.906
	3	0.748	0.128	0.755	0.356	0.922	0.565

Insgesamt zeigt sich jedoch für die *total number of saccades* aus Sicht der deskriptiven Statistik kein nennenswerter Unterschied zwischen den einzelnen Phasen der Erfahrungsstufen.
Die Ergebnisse werden erneut durch einen Boxplot veranschaulicht. Dieser findet sich in Abbildung 5.17.

Phasenbasierte Betrachtung der *average number of saccades*

Die phasenbasierte Betrachtung der *average number of saccades* fördert die selben Ergebnisse zutage, wie die zuvor beschriebene *total number of saccades*. Die Friedman-Tests zeigen keine signifikanten Unterschiede zwischen den einzelnen Phasen (siehe Tabelle 5.30) auf.
Hinsichtlich der begleitenden deskriptiven Statistik (siehe Tabelle 5.31) verhält sich die *average number of saccades* ebenfalls identisch zur *total number of saccades*: Die Expertinnen und Experten zeigen in ihren jeweiligen Phasen etwas weniger Saccaden, als die Novizinnen und Novizen. Dennoch zeigt sich zwischen den Phasen kein signifikanter Unterschied.
Ein Boxplot illustriert erneut die Ergebnisse und es lässt sich auch an diesem erkennen, dass die *average number of saccades* für beide Gruppen in den Phasen relativ konstant bleibt. Der Boxplot findet sich in Abbildung 5.18.

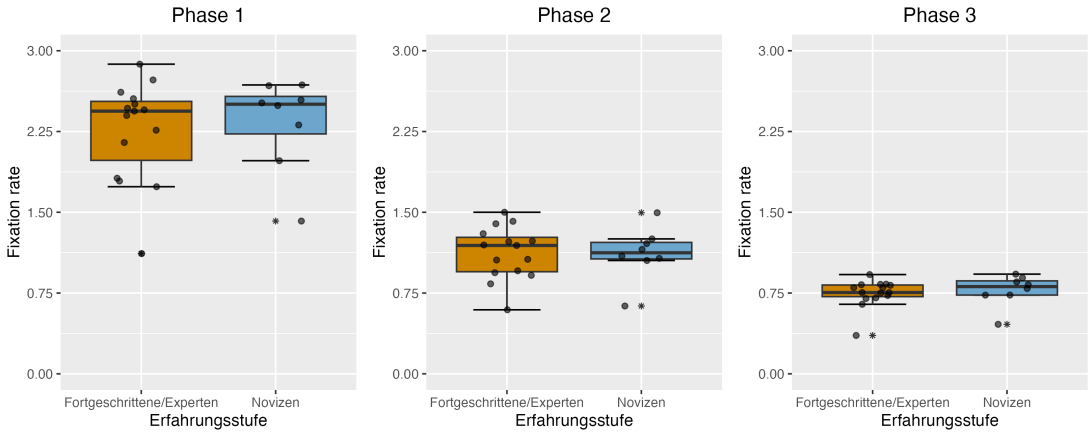

Abbildung 5.16: Betrachtung der phasenbasierten Daten zur *fixation rate*

Tabelle 5.28: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of saccades

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=0.783; p=0.676 $FT_{(Experten)}$: $\chi^2(2)$=0.750; p=0.687 $FT_{(Novizen)}$: $\chi^2(2)$=0.400; p=0.819			

Phasenbasierte Betrachtung der *total number of visits on erroneous lines*

Den Abschluss der phasenbasierten Betrachtung stellen die Eye-Tracking-Metriken dar, welche auf AOIs beruhen. Hier wird mit der Analyse der *total number of visits on erroneous lines* begonnen. Für diese lassen sich durch die Friedman-Tests erneut Unterschiede feststellen (siehe Tabelle 5.32).

Übergreifend über beide Gruppen betrachtet ($FT_{(Gesamt)}$: $\chi^2(2)$=27.714; p=0.000) machen sich Unterschiede zwischen Phase 1 (M=19.087; SD=15.213; Mdn=17.000) und Phase 2 (M=39.783; SD=23.454; Mdn=32.000) (p=0.000), sowie zwischen Phase 1 und Phase 3 (M=45.870; SD=29.575; Mdn=44.000) (p=0.000) bemerkbar. Der Post-hoc-Test bestätigt deren statistische Signifikanz.

Auf Seiten der Expertinnen und Experten zeichnet sich ein ähnliches Bild ab: Der Friedman-Test ($FT_{(Experten)}$: $\chi^2(2)$=10.903; p=0.004) bestätigt das Vorhandensein von Unterschieden und der Post-hoc-Test lokalisiert diese ebenfalls zwischen Phase 1 (M=13.625; SD=8.733; Mdn=14.000) (p=0.036) und Phase 2 (M=37.500; SD=23.982; Mdn=28.000) (p=0.024), sowie zwischen Phase 1 und Phase 3 (M=39.750; SD=24.904; Mdn=37.000).

Auch für die Gruppe der Novizinnen und Novizen können durch den Friedman-Test ($FT_{(Novizen)}$: $\chi^2(2)$=16.933; p=0.000) an den selben Stellen Unterschiede loksalisiert

Tabelle 5.29: Deskriptive Statistik zur phasenbasierten Betrachtung der *total number of saccades*

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	524.304	306.228	414.000	205.000	1400.000	1195.000
	2	530.913	288.577	448.000	242.000	1380.000	1138.000
	3	539.870	268.216	493.000	232.000	1249.000	1017.000
Experten	1	420.250	209.209	379.000	205.000	880.000	675.000
	2	422.625	189.336	383.500	242.000	826.000	584.000
	3	429.000	200.767	352.500	232.000	839.000	607.000
Novizen	1	579.800	340.573	414.000	277.000	1400.000	1123.000
	2	588.667	320.413	470.000	273.000	1380.000	1107.000
	3	599.000	286.574	505.000	261.000	1249.000	988.000

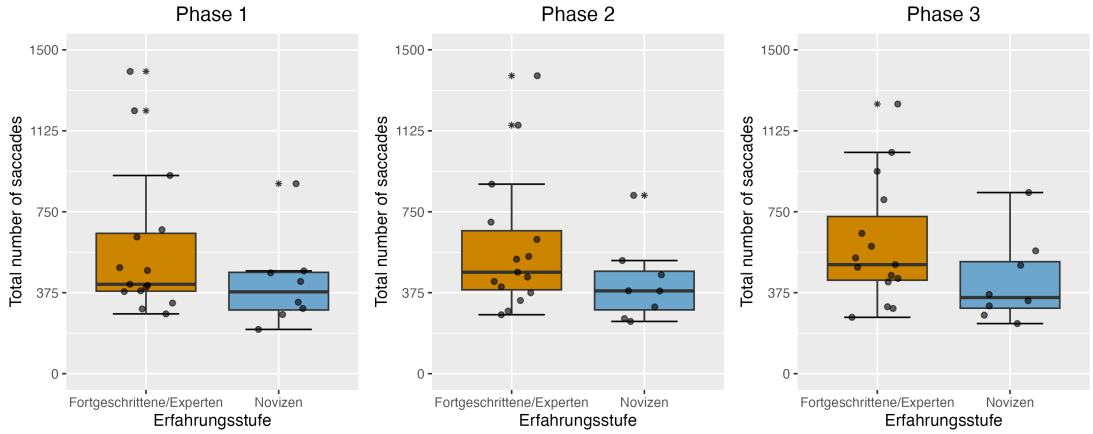

Abbildung 5.17: Betrachtung der phasenbasierten Daten zur *total number of saccades*

werden. Phase 1 (M=22.000; SD=17.304; Mdn=17.000) unterscheidet sich signifikant (p=0.002) von Phase 2 (M=41.000; SD=23.922; Mdn=34.000) und Phase 3 (M=49.133; SD=32.133; Mdn=44.000) (p=0.003).
Eine detaillierte Darstellung der deskriptiven Statistik kann Tabelle 5.33 entnommen werden.
Die grafisch aufbereiteten deskriptiven Ergebnisse finden sich auch für diese Metrik in einem Boxplot. Dieser wird in der Abbildung 5.19 dargestellt.

Phasenbasierte Betrachtung der *average number of visits on erroneous lines*

Die *average number of visits on erroneous lines* verhält sich identisch zur zuvor beschriebenen *total number of visits on erroneous lines*. Die Friedman-Tests (siehe Tabelle 5.34) schlagen erneut an und weisen auf signifikante Unterschiede hin.
Erneut zeigen sich für die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)=27.714$; p=0.000) signifikante Unterschiede. Diese befinden sich laut der Post-hoc-Tests zwischen der ersten (M=3.181; SD=2.536; Mdn= 2.833) und zweiten (M=6.630; SD=3.909; Mdn=5.333) (p=0.000), sowie zwischen der ersten und dritten Phase (M=7.645; SD=4.931; Mdn=7.333) (p=0.000).

Tabelle 5.30: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average number of saccades

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
	$FT_{(Gesamt)}$: $\chi^2(2)$=0.783; p=0.676 $FT_{(Experten)}$: $\chi^2(2)$=0.750; p=0.687 $FT_{(Novizen)}$: $\chi^2(2)$=0.400; p=0.819		

Tabelle 5.31: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of saccades

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	87.384	51.038	69.000	34.167	233.333	199.167
	2	88.486	48.096	74.667	40.333	230.000	189.667
	3	89.978	44.703	82.167	38.667	208.167	169.500
Experten	1	70.042	34.868	63.167	34.167	146.667	112.500
	2	70.438	31.556	63.917	40.333	137.667	97.333
	3	71.500	33.461	58.750	38.667	139.833	101.167
Novizen	1	96.633	56.762	69.000	46.167	233.333	187.167
	2	98.111	53.402	78.333	45.500	230.000	184.500
	3	99.833	47.762	84.167	43.500	208.167	164.667

Die Betrachtung der Expertinnen und Experten zeigt das gleiche Resultat ($FT_{(Experten)}$: $\chi^2(2)$=10.903; p=0.004). Die Post-hoc-Tests lokalisieren die Unterschiede ebenfalls zwischen Phase 1 (M=2.271; SD=1.456; Mdn=2.333) und 2 (M=6.250; SD=3.997; Mdn=4.667) (p=0.036) und zwischen Phase 1 und 3 (M=6.625; SD=4.151; Mdn=6.167) (p=0.024).
Für die Novizinnen und Novizen können ebenfalls signifikante Unterschiede beobachtet werden ($FT_{(Novizen)}$: $\chi^2(2)$=16.933; p=0.000). Diese liegen erneut zwischen Phase 1 (M=3.667; SD=2.884; Mdn=2.833) und 2 (M=6.833; SD=3.987; Mdn=5.667) (p=0.002), sowie zwischen Phase 1 und 3 (M=8.189; SD=5.356; Mdn=7.333) (p=0.003).
Weitere Informationen bezüglich der deskriptiven Statistik zur phasenbasierten Betrachtung der *average number of visits on erroneous lines* kann der nachfolgenden Tabelle 5.35 entnommen werden.
Eine grafische Aufbereitung der deskriptiven Statistik findet sich in eines Boxplots in Abbildung 5.20.

Phasenbasierte Betrachtung der *total dwell time on erroneous lines in [ms]*

Bei der phasenbasierten Betrachtung der *total dwell time on erroneous lines in [ms]* zeigen sich ebenfalls Unterschiede zwischen den Phasen. Die durchgeführten Friedman-

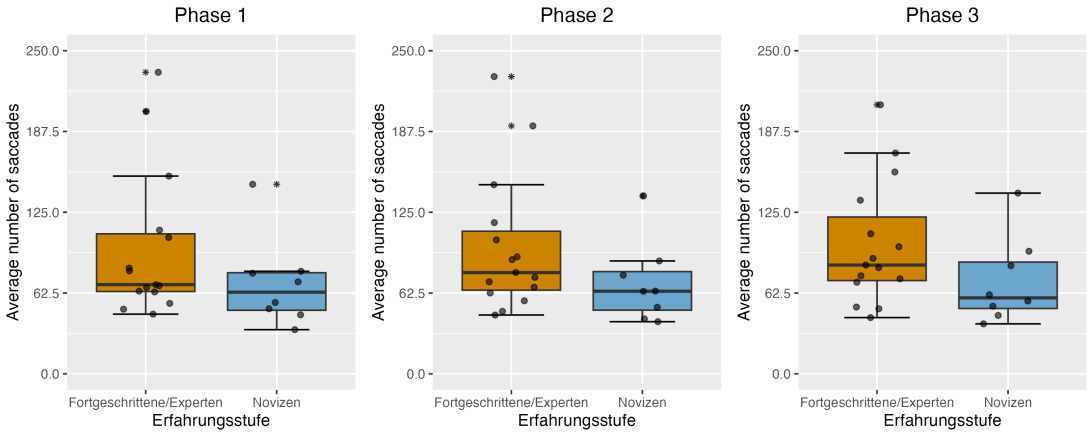

Abbildung 5.18: Betrachtung der phasenbasierten Daten zur *average number of saccades*

Tabelle 5.32: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of visits on erroneous lines

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.000)	Unterschied (p=0.000)	Kein Unterschied (p=1.000)
Experten	Unterschied (p=0.036)	Unterschied (p=0.024)	Kein Unterschied (p=1.000)
Novizen	Unterschied (p=0.002)	Unterschied (p=0.003)	Kein Unterschied (p=1.000)
	$FT_{(Gesamt)}$: $\chi^2(2)$=27.714; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=10.903; p=0.004 $FT_{(Novizen)}$: $\chi^2(2)$=16.933; p=0.000		

Tests liefern diesbezüglich signifikante Ergebnisse für die gesamte Stichprobe, sowie für die Gruppe der Novizen (siehe Tabelle 5.36).

Für die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)$=15.217; p=0.000) zeigen die Post-hoc-Tests zwei signifikante Unterschiede an. Diese befinden sich zwischen der ersten (M=9188.523; SD=9683.909; Mdn=7931.694) und zweiten (M=17163.714; SD=12831.979; Mdn=11179.463) (p=0.020), sowie zwischen der ersten und dritten Phase (M=18251.486; SD=14585.632; Mdn=13758.456) (p=0.000).

Eine Betrachtung der Gruppe der Expertinnen und Experten führt zu keinem signifikanten Ergebnis. Der Fridman-Test ($FT_{(Experten)}$: $\chi^2(2)$=6.750; p=0.034) deutet zwar das Vorhandensein von Unterschieden an, diese können aber durch die Post-hoc-Tests nicht identifiziert werden.

In Bezug auf die Novizen ($FT_{(Novizen)}$: $\chi^2(2)$=8.933; p=0.011) ergibt sich ein signifikanter Unterschied. Dieser befindet sich zwischen der ersten (M=11183.376; SD=11371.669; Mdn=9875.865) und dritten Phase (M=18411.896; SD=14773.453; Mdn=14727.194) (p=0.021).

Tabelle 5.37 bietet eine Übersicht über die gesamte deskriptive Statistik zur *total dwell time on erroneous lines in [ms]*. Diese wird weiterhin in Abbildung 5.21 als Boxplot dargestellt.

Tabelle 5.33: Deskriptive Statistik zur phasenbasierten Betrachtung der total number of visits on erroneous lines

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	19.087	15.213	17.000	1.000	72.000	71.000
	2	39.783	23.454	32.000	5.000	94.000	89.000
	3	45.870	29.585	44.000	2.000	133.000	131.000
Experten	1	13.625	8.733	14.000	2.000	27.000	25.000
	2	37.500	23.982	28.000	19.000	91.000	72.000
	3	39.750	24.904	37.000	9.000	83.000	74.000
Novizen	1	22.000	17.304	17.000	1.000	72.000	71.000
	2	41.000	23.922	34.000	5.000	94.000	89.000
	3	49.133	32.133	44.000	2.000	133.000	131.000

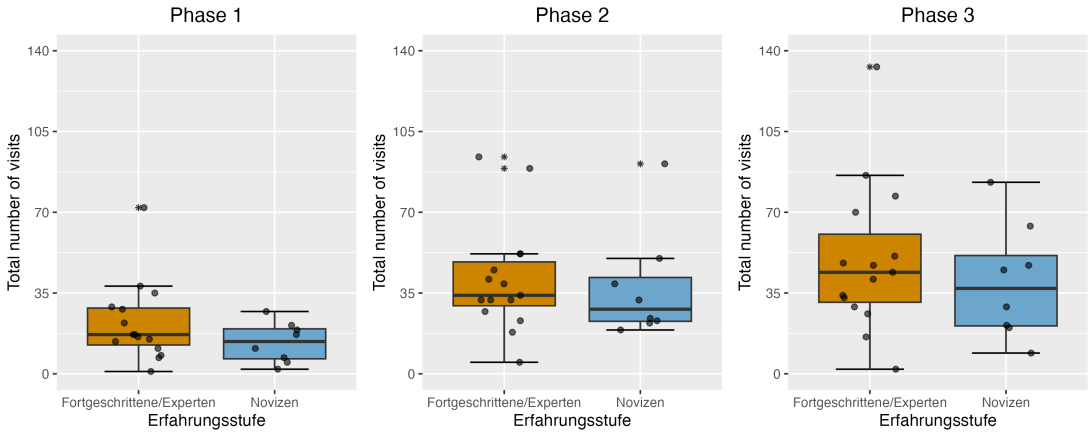

Abbildung 5.19: Betrachtung der phasenbasierten Daten zur *total number of visits on erroneous lines*

Phasenbasierte Betrachtung der *average dwell time on erroneous lines in [ms]*

Die letzte phasenbasierte Metrik, welche im Kontext dieser Studie analysiert ist, ist die *average dwell time on erroneous lines in [ms]*. Diese verhält sich identisch zur zuvor beschriebenen *total dwell tiome on erroneous lines in [ms]*. Die Friedman-Tests deuten auch für diese das Vorhandensein von signifikanten Unterschieden zwischen den Phasen an (siehe Tabelle 5.38.

Erneut zeigen sich für die gesamte Stichprobe signifikante Unterschiede ($FT_{(Gesamt)}$: $\chi^2(2)=15.217$; p=0.000). Diese befinden sich auch im Falle der phasenbasierten Betrachtung der *average dwell time on erroneous lines* zwischen den Phasen 1 (M1531.420; SD=1613.985; Mdn=1321.949) und 2 (M=2860.619; SD=2138.663; Mdn=1863.244) (p=0.020), sowie zwischen Phase 1 und 3 (M=3041.914; SD=2430.939; Mdn=2293.077) (p=0.001).

Bezüglich der Expertinnen und Experten kann trotz eines signifikanten Friedman-Tests ($FT_{(Experten)}$: $\chi^2(2)=6.750$; p=0.034) kein signifikanter Unterschied zwischen den Phasen lokalisiert werden.

Die Gruppe der Novizinnen und Novizen weist ebenfalls einen statistisch signifikanten Friedman-Test auf ($FT_{(Novizen)}$: $\chi^2(2)=8.933$; p=0.011). Durch die Post-hoc-Tests

Tabelle 5.34: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der *average number of visits on erroneous lines*

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.000)	Unterschied (p=0.000)	Kein Unterschied (p=1.000)
Experten	Unterschied (p=0.036)	Unterschied (p=0.024)	Kein Unterschied (p=1.000)
Novizen	Unterschied (p=0.002)	Unterschied (p=0.003)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=27.714; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=10.903; p=0.004 $FT_{(Novizen)}$: $\chi^2(2)$=16.933; p=0.000			

Tabelle 5.35: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of visits on erroneous lines

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	3.181	2.536	2.833	0.167	12.000	11.833
	2	6.630	3.909	5.333	0.833	15.667	14.833
	3	7.645	4.931	7.333	0.333	22.167	21.833
Experten	1	2.271	1.456	2.333	0.333	4.500	4.167
	2	6.250	3.997	4.667	3.167	15.167	12.000
	3	6.625	4.151	6.167	1.500	13.833	12.333
Novizen	1	3.667	2.884	2.833	0.167	12.000	11.833
	2	6.833	3.987	5.667	0.833	15.667	14.833
	3	8.189	5.356	7.333	0.333	22.167	21.833

kann dieser zwischen den Phasen 1 (M=1863.896; SD=1895.278; Mdn=1645.977) und 3 (M=3068.649; SD=2462.242; Mdn=2454.532) (p=0.021) lokalisiert werden.
Einen Gesamtüberblick über die deskriptive Statistik bietet die Tabelle 5.39. Diese wird auch in Abbildung 5.22 illustriert.

5.4 Diskussion

Nachfolgend sollen sowohl die Ergebnisse der zuvor beschriebenen allgemeinen Eye-Tracking-Metriken (siehe Absatz 5.4.1), als auch deren phasenbasierte Betrachtung (siehe Absatz 5.4.2) diskutiert und durch Vergleiche mit ähnlichen Studien eingeordnet werden.

5.4.1 Diskussion der allgemeinen Eye-Tracking-Metriken

Die Ergebnisse der deskriptiven Statistik zu den allgemeinen Eye-Tracking-Metriken wird nachfolgend diskutiert. Dabei werden die Erkenntnisse dieser Dissertation mit den Resultaten von anderen Forschern verglichen.
Bezüglich der *total* und *average number of fixations* ergeben sich marginale Unterschie-

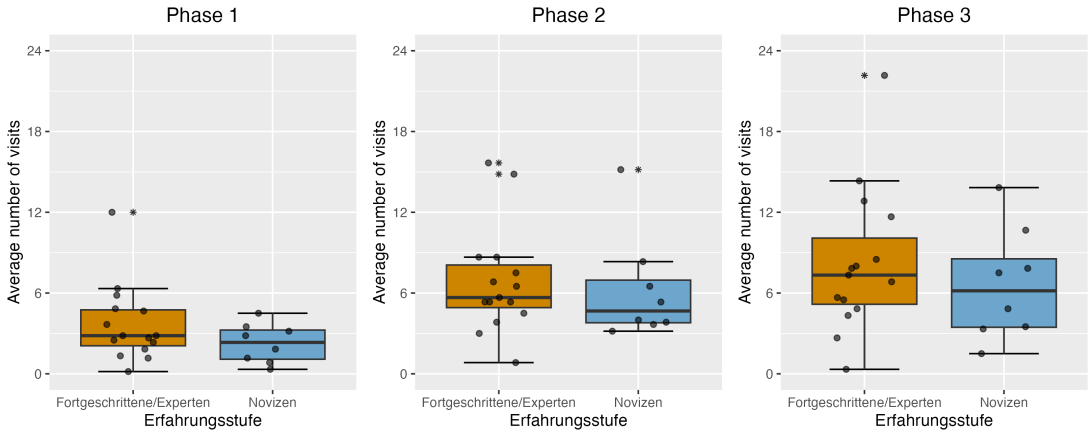

Abbildung 5.20: Betrachtung der phasenbasierten Daten zur *average number of visits on erroneous lines*

Tabelle 5.36: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total dwell time on erroneous lines in [ms]

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.020)	Unterschied (p=0.001)	Kein Unterschied (p=1.000)
Experten	Kein Unterschied (p=0.147)	Kein Unterschied (p=0.147)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=0.268)	Unterschied (p=0.021)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=15.217; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=6.750; p=0.034 $FT_{(Novizen)}$: $\chi^2(2)$=8.933; p=0.011			

de zwischen den Experten und Novizen, die jedoch keine statistische Signifikanz erreichen. Auch die kalkulierten Korrelationen bezüglich möglicher Zusammenhänge zwischen diesen Metriken und der *allgemeinen* und *professionellen Programmiererfahrung*, sowie der durchgeführte Mann-Whitney-U-Test erzielen keine signifikanten Ergebnisse und zeigen lediglich schwache Effekte. Vor dem Hintergrund der *holistic models of image perception* (Nodine & Kundel, 1987; Kundel et al., 2007; Sheridan & Reingold, 2017; Swensson, 1980) betrachtet, zeigt sich die Tendenz, dass die Experten die Codebeispiele mit weniger *fixations* begutachten, was eine effektivere Vorgehensweise implizieren könnte.

Ein ähnliches Bild zeigt sich bei der Betrachtung der *total* und *average fixation duration in [ms]*. Auch bei diesen Metriken zeigen sich nur schwache Unterschiede zwischen den Experten und Novizen. Weder die untersuchten Korrelationskoeffizienten, noch die die Mann-Whitney-U-Tests zeigen signifikante Unterschiede an. Bezüglich der *total fixation duration in [ms]* ergibt sich bei der Betrachtung der Mediane eine Auffälligkeit: Der Median für die Experten fällt mit einem Wert von 402620.885 ms deutlich niedriger aus, als für die Novizen. Im Sinne der *holistic models of image perception* (Kundel & La Follette, 1972; Nodine & Kundel, 1987; Kundel et al., 2007; Swensson, 1980; Sheridan & Reingold, 2017), als auch vom Standpunkt der Erforschung von visueller Expertise, wäre bei diesen Metriken, insbesondere bei der *average fixati-*

Tabelle 5.37: Deskriptive Statistik zur phasenbasierten Betrachtung der total dwell time on erroneous lines in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	9188.523	9683.909	7931.694	240.103	48961.153	48721.05
	2	17163.714	12831.979	11179.463	1623.601	43637.668	42014.067
	3	18251.486	14585.632	13758.465	535.921	53424.418	52888.497
Experten	1	5448.173	3403.709	5551.747	851.914	9375.221	8523.307
	2	16504.883	13289.659	9623.601	5679.482	37453.524	31774.042
	3	17950.719	15229.746	10786.924	2379.867	45240.828	42860.961
Novizen	1	11183.376	11371.669	9875.865	240.103	48961.153	48721.05
	2	17515.09	13040.825	11690.972	1623.601	43637.668	42014.067
	3	18411.896	14773.453	14727.194	535.921	53424.418	52888.497

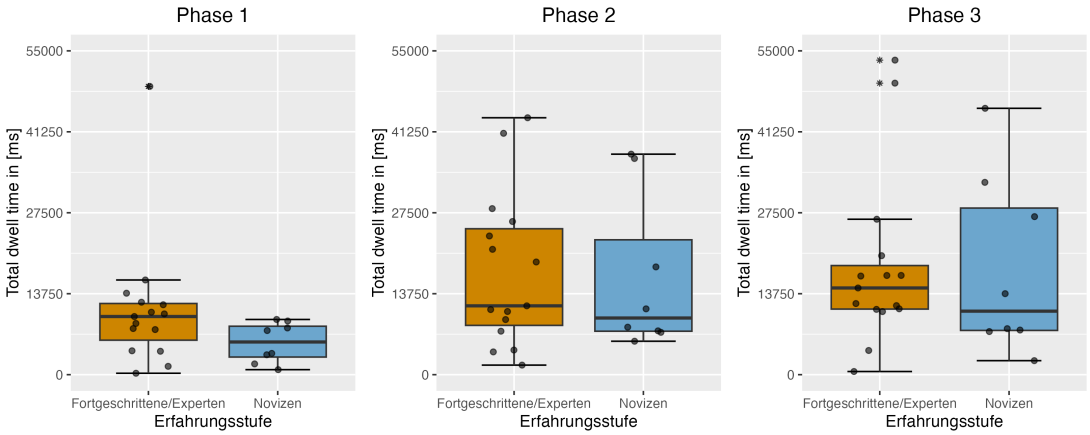

Abbildung 5.21: Betrachtung der phasenbasierten Daten zur *total dwell time on erroneous lines in [ms]*

on duration in [ms] mit signifikanteren Ergebnissen und deutlicheren Gruppenunterschieden zu rechnen gewesen. Diese können jedoch nicht nachgewiesen werden. Die Experten performen zwar hinsichtlich beider Metriken etwas besser, jedoch nicht so ausgeprägt, wie es auf Basis der theoretischen Grundlagen zu erwarten wäre.
Die Betrachtung der *fixation rate* führt ebenfalls zu keinen statistisch signifikanten Ergebnissen. Mit einem Mittelwert von 1.408 *fixation/sec* (SD=0.254; Mdn=1.467) für die Experten und einem Mittelwert von 1.377 *fixation/sec* (SD=0.271; Mdn=1.453) zeigt sich für die erfahreneren Versuchsperson die Tendenz, die Codebeispiele im Sinne eines *Scan* (Uwano et al., 2006; Nodine & Kundel, 1987; Swensson, 1980) etwas intensiver abzutasten. Obwohl nicht signifikant, zeigt sich dennoch ein auffälliger Korrelationskoeffizienten für den Zusammenhang zwischen der *fixation rate* und der *professionellen Programmiererfahrung*. Dieser beläuft sich auf r_{BP}=0.234 (p=0.283). Interessant ist diesbezüglich, dass sich kein Zusammenhang mit der *allgemeinen Programmiererfahrung* ergibt, bzw. dieser sehr schwach ausfällt (r_{BP}=0.008; p=0.972). Im Sinne der Expertiseforschung (Billett et al., 2018) scheint sich hier zumindest anzudeuten, dass das Ansammeln von Berufserfahrung in dieser Domäne einen Beitrag zum Aufbau von Expertise leistet.

Tabelle 5.38: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der *average dwell time on erroneous lines in [ms]*

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.020)	Unterschied (p=0.001)	Kein Unterschied (p=1.000)
Experten	Kein Unterschied (p=0.147)	Kein Unterschied (p=0.147)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=0.268)	Unterschied (p=0.021)	Kein Unterschied (p=1.000)
	$FT_{(Gesamt)}$: $\chi^2(2)$=15.217; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=6.750; p=0.034 $FT_{(Novizen)}$: $\chi^2(2)$=8.933; p=0.011		

Tabelle 5.39: Deskriptive Statistik zur phasenbasierten Betrachtung der average dwell time on erroneous lines in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	1531.420	1613.985	1321.949	40.017	8160.192	8120.175
	2	2860.619	2138.663	1863.244	270.600	7272.945	7002.344
	3	3041.914	2430.939	2293.077	89.320	8904.070	8814.750
Experten	1	908.029	567.285	925.291	141.986	1562.537	1420.551
	2	2750.814	2214.943	1603.934	946.580	6242.254	5295.674
	3	2991.787	2538.291	1797.821	396.645	7540.138	7143.493
Novizen	1	1863.896	1895.278	1645.977	40.017	8160.192	8120.175
	2	2919.182	2173.471	1948.495	270.600	7272.945	7002.344
	3	3068.649	2462.242	2454.532	89.320	8904.070	8814.750

Obwohl sich bei der Betrachtung der *total* und *average number of saccades* ebenfalls keine signifikanten Ergebnisse zeigen, präsentieren sich diese Metriken in Bezug auf die Verwendung der *holistic models of image perception*, als auch auf die Expertiseforschung interessanter. Beide Metriken fallen für die Experten deutlich niedriger aus und zeigen für diese Gruppe eine geringere Standardabweichung. Auf Basis der *holistic models of image perception* betrachtet, kann davon ausgegangen werden, dass die Resultate auf eine effizientere Strategie der Experten hindeutet, welche vermutlich mit einem größeren Erkenntnisgewinn durch das *global viewing* bzw. den initialen Scan des Codes eingergeht (Begel & Vrzakova, 2018; Busjahn et al., 2015; Nodine & Kundel, 1987; Sharif et al., 2012; Swensson, 1980; Uwano et al., 2006).

Ähnliche Bilder zeigen sich auch bei den AOI-basierten Metriken. Bei der *total* und *average number of visits on erroneous lines* zeigen sich erneut marginale, nicht signifikante Ergebnisse, die darauf hindeuten, dass die Experten Fehler mit weniger visuellem Aufwand im Sinne von weniger *fixations* auf die fehlerhafte Zeile identifizieren können. Der exakt gleiche Trend zeigt sich für die *average* und *total dwell time on erroneous lines in [ms]*. Trotz nicht signifikanter Resultate zeigt sich, dass auch eine kürzere Betrachtungsdauer der fehlerhaften Zeile für die Experten ausreichend ist, um einen Fehler zu identifizieren. Dies könnte als Hinweis für den Einfluss von do-

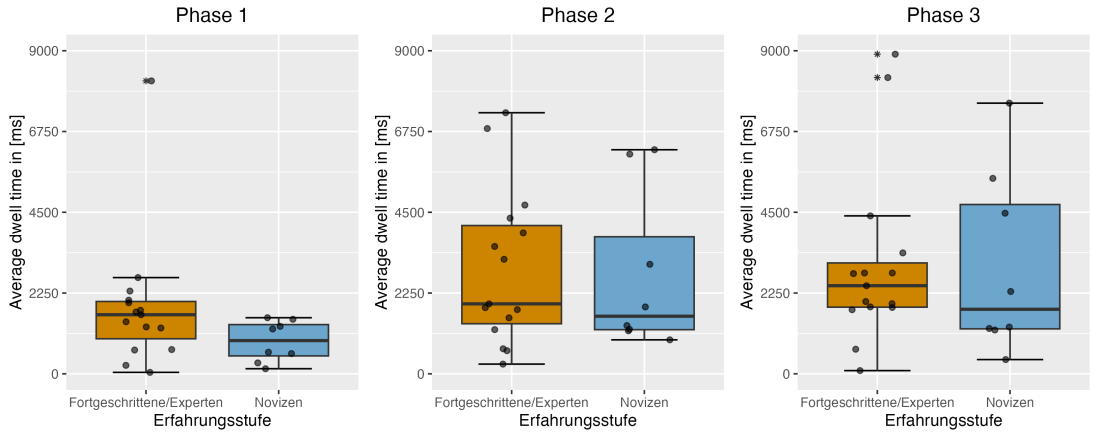

Abbildung 5.22: Betrachtung der phasenbasierten Daten zur *average dwell time on erroneous lines*

mänenspezifischer Erfahrung gewertet werden, mit der auch Wissen über Strukturen des Quellcodes, vulnerable Stellen und spezielle Fehler einhergeht (Billett et al., 2018; Ericsson & Towne, 2010; Gruber, 2007). Interessant ist in diesem Fall auch, dass die Ergebnisse konträr zu den *holistic models of image perception* verlaufen, welche eher naheliegen, dass Auffälligkeiten von Experten häufiger und länger betrachtet werden (Sheridan & Reingold, 2017). Es muss allerdings auch angemerkt werden, dass diese Vermutung aus der Radiologie (Gegenfurtner et al., 2011; Kok, 2016; Sheridan & Reingold, 2017) stammt und nicht belegbar ist, ob sich alle Annahmen problemlos auf Code Reviews übertragen lassen.

5.4.2 Diskussion der phasenbasierten Eye-Tracking-Metriken

Anschließend an die Diskussion der allgemeinen Eye-tracking-Metriken sollen nachfolgend die Ergebnisse der phasenbasierten Analyse besprochen und verglichen werden. Die Vorgehensweise, welche vorrangig auf der Nutzung der Firedman-Tests beruht konnte ausgeprägtere Unterschiede zwischen den Experten und Novizen identifizieren und diese mit den Phasen des Reviews in Verbindung setzen. Diskussionswürdige Auffälligkeiten ergaben sich bei der *total* und *average fixation duration*, der *fixation rate*, sowie bei der *total* und *average number of visits on erroneous lines*.

Bei der *total fixation duration* zeigt sich für beide Gruppen, dass diese im Verlauf des Experiments geringer wird. Dies wird auch durch signifikante Friedman-Tests verdeutlicht. Obwohl die Unterschiede gering wirken, erreichen sie dennoch statistische Signifikanz. Das Abfallen der *total fixation duration* könnte darauf zurückzuführen sein, dass die Codebeispiele im Verlauf des Experiments als mentales Modell aufgebaut werden und Details besser erinnert werden (Hauser, Reuter et al., 2018).

Hinsichtlich der *average fixation duration* gestalten sich weniger die Unterschiede zwischen den Phasen interessant, sondern die Unterschiede zwischen den Experten und Novizen. Die Mittelwerten für die Experten (Phase 1: M=394.733; SD=139.470; Phase 2: M=397.315; SD=161.577; Phase 3: M=374.078; SD=148.574) fallen deutlich

höher aus als für die Novizen (Phase 1: M=89.022; SD=33.230; Phase 2: M=88.022; SD=32.828; Phase 3: M=90.667; SD=36.584). Eine Erklärung auf Basis der *holistic models of image perception* legt nahe, dass es sich bei den *fixations* der Experten vermutlich eher um eine fokale und damit längere bzw. intensivere Betrachtung der Codebeispiele handelt. Dies könnte implizieren, dass auf Seiten der Experten weniger *fixations* benötigt werden um sich einen Überblick über die Stimuli zu verschaffen und direkt sehr viel früher als bei den Novizen mit der Suche nach Fehlern begonnen wird (Nodine & Kundel, 1987; Kok, 2016; Kundel et al., 2007; Sheridan & Reingold, 2017; Swensson, 1980). Hier scheint auch das domänenspezifische Wissen von Bedeutung zu sein und einen Einfluss auf die gezeigte Strategie auszuüben (Ericsson & Towne, 2010; Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Gegenfurtner, 2020).

Die Unterschiede bei der *fixation rate* fallen bei einem Vergleich der Gruppen eher gering aus. Für die Experten und Novizen lässt sich beobachten, dass die Anzahl von Fixierungen pro Sekunde im Verlauf des Experiments sinkt. Dies deutet im Kontext der *holistic models of image perception* darauf hin, dass die Vorgehensweise von beiden Gruppen durch einen initialen Scan (Nodine & Kundel, 1987; Sharif et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980; Uwano et al., 2006) gekennzeichnet ist und im Verlauf des Reviews eine genauere Betrachtung der Stimuli erfolgt. Somit finden sich bei den Code Reviews ebenfalls Anzeichen für *global* und *focal viewing*, die typisch für die *holistic models of image perception* sind (Kundel et al., 2007; Nodine & Kundel, 1987; Swensson, 1980). Auch die Ergebnisse aus der Studie von Begel und Vrzakova (2018), sowie die Arbeit von Peachock et al. (2017) deuten vor einem anderen theoretischen Grundgerüst auf ähnliche Ergebnisse hin.

Bei der *total* und *average number of visits on erroneous lines* zeigt sich bei beiden Metriken ein ähnliches Bild: Die Friedman-Tests belegen signifikante Unterschiede zwischen den Phasen und auch die Experten und Novizen unterscheiden sich hinsichtlich ihrer Vorgehensweise. Bei beiden Metriken und bei beiden Gruppen fällt die erste Phase am geringsten aus. In Phase 2 und 3 steigt die Anzahl der *visits on erroneous lines* an, was drauf hindeutet, dass in diesen die fehlerhaften Zeilen mehr Aufmerksamkeit erhalten und häufiger betrachtet werden. Im Kontext der *holistic models of image perception* deutet sich somit an, dass eine Form des *global viewings* nach einem initialen Scan praktiziert wird (Nodine & Kundel, 1987; Sharif et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980; Uwano et al., 2006). Bei genauerer Betrachtung der Ergebnisse zeigt sich auch, dass die Experten bei beiden Metriken eine geringere Anzahl an *visits* aufweisen als die Novizen. Dies könnte ebenfalls auf den Einfluss von domänenspezifischen Vorwissen hinweisen, welches sich positiv auf die Fehlererkennung auswirkt (Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Kok, 2016; Sheridan & Reingold, 2017; Swensson, 1980). Bezogen auf die Experten ist es ebenfalls noch interessant, dass sich deren *number of visits* ab der zweiten Phase als relativ konstant erweist und sich gegenüber der dritten Phase nur minimal verändert. Auf Seiten der Novizen ist hier nochmals ein Anstieg zu beobachte.

Dies könnte implizieren, dass die Bewertung der Fehler für die Novizen aufwändiger ist, mehr Absicherung bedarf und insgesamt länger dauert als für die Experten. Auch in diese Fall würde das domänenspezifische Vorwissen und die angepasste Betrachtungsstrategie eine geeignete Grundlage für eine Erklärung darstellen (Begel & Vrzakova, 2018; Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Kok, 2016; Kundel et al., 2007; Nodine & Kundel, 1987; Peachock et al., 2017; Sharafi et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980).

5.5 Einschränkungen

Hinsichtlich möglicher Einschränkungen stellt sich im Falle dieser Studie vor allem die bereits angesprochene Stichprobengröße und deren unausgewogene Unterteilung in zwei Erfahrungsstufen als problematisch dar. Diese Problematik trägt auch dazu bei, dass die Verteilungen der untersuchten Metriken keiner Normalverteilung entsprechen. Aufgrund der nichtvorhandenen Normalverteilung werden beispielsweise auch die Voraussetzungen für die Berechnung des Bravais-Pearson-Korrelationskoeffizienten verletzt und die Durchführung der statistischen Analysen erschwert. Die eingesetzten Verfahren werden jedoch als robust genug erachtet um im Kontext dieser Dissertation verwendet zu werden (Bortz & Schuster, 2010; Döring & Bortz, 2016; Field, Miles & Field, 2012).

Weiterhin stellen die verwendeten Codebeispiele eine Einschränkung dar. Diese sind zwar eine möglichst genaue Replik der Originalstudie von Uwano et al. (2006), stellen aber aus Sicht des Software Engineerings ein Problem dar: Entwickler arbeiten im Regelfall nicht mit derart kurzen Quellcodes (Broy, 2006; Sommerville, 2012; Sharif, Shaffer, Wise & Maletic, 2016). Diese sind in der Realität deutlich länger und können in der heutigen Zeit einen Umfang von mehreren tausend oder gar Millionen Codezeilen aufweisen (Broy, 2006). Insofern stellt sich die Frage, wie sehr die verwendeten Stimuli wirklich die Versuchspersonen (vor allem erfahrene Programmiererinnen und Programmierer) dazu animieren ihr volles Fähigkeitsniveau auszureizen?

Ähnlich verhält es sich zum Setup des Experiments. Dieses wurde basierend auf Uwano et al. (2006) in einer Eye-Tracking-Software aufgesetzt. Entwickler würden Code jedoch in einer speziellen Entwicklungsumgebung (auch als *integrated development enviroment* oder kurz als *IDE* bekannt) betrachten. Diese ermöglicht eine deutlich intensivere Auseinandersetzung und Begutachtung mit dem Code, als es in der verwendeten Eye-Tracking-Software von SMI möglich ist (SensoMotoric Instruments, 2017). Auch hier stellt sich die Frage, wie stark die Abweichung zur Realität ist.

5.6 Überprüfung der Hypothesen

Auf die Diskussion der Ergebnisse folgend, wird unter Berücksichtigung der zuvor genannten Einschränkungen nachfolgenden überprüft, welche der in Kapitel 4 formulierten Hypothesen (siehe Absatz 4.2 bestätigt oder verworfen werden kön-

nen. Diesbezüglich wird ebenfalls eine zweistufige Vorgehensweise angewandt: In einem ersten Schritt werden die allgemeinen Hypothesen zu den verschiedenen Eye-Tracking-Metriken überprüft (siehe Absatz 5.6.1), darauf folgend werden deren phasenbasierte Varianten (siehe Absatz 5.6.2) betrachtet.

5.6.1 Überprüfung der allgemeinen Hypothesen

Die Überprüfung der allgemeinen Hypothesen fördert lediglich einen signifikanten Unterschied für die *Performance* zutage (siehe Absatz 5.4.1). Tabelle 5.40 fasst unter Berücksichtigung der limitierenden Faktoren (sieh 5.5) die Resultate zusammen und stellt dar, welche Hypothesen verworfen oder beibehalten werden.

Tabelle 5.40: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Performance	H1	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten. Experten finden signifikant mehr Fehler.
Total number of fixations	H0	Die total number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es kann jedoch beobachtet werden, dass Experten insgesamt weniger fixations für die Durchführung des Reviews benötigen.
Average number of fixations	H0	Die average number of fixations während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigt sich jedoch, dass diese für die Experten geringer ausfällt.
Total fixation duration	H0	Die total fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.

Tabelle 5.40: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Average fixation duration	Ho	Die average fixation duration während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.
Fixation rate	Ho	Die fixation rate während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen den Novizen und Experten. Die Experten weisen einen niedrigeren Wert für die fixation rate auf.
Total number of saccades	Ho	Die total number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Für die Experten zeigt sich ein insgesamt niedrigerer Wert für die total number of saccades.
Average number of saccades	Ho	Die average number of saccades während eines Code Reviews in der Programmiersprache C unterscheidet sich nicht signifikant zwischen Novizen und Experten. Bei den Experten zeigt sich ein niedrigerer Wert für die average number of saccades.
Total number of visits on erroneous lines	Ho	Die Die total number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeigt sich ein im Vergleich zu den Novizen geringerer Median.

Tabelle 5.40: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Average number of visits on erroneous lines	Ho	Die average number of visits on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen geringerer Median.
Total dwell time on erroneous lines	Ho	Die total dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen geringerer Median.
Average dwell time on erroneous lines	Ho	Die average dwell time on erroneous lines während eines Code Reviews in der Programmiersprache C unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen geringerer Median.

5.6.2 Überprüfung der phasenbasierten Hypothesen

Im Vergleich zu den zuvor in Absatz 5.6.1 behandelten allgemeinen Hypothesen zeigen sich für die phasenbasierte Betrachtung signifikante Unterschiede. Die nachfolgende Tabelle 5.41 stellt unter Berücksichtigung der Diskussion (siehe Absatz 5.4) und der genannten Einschränkungen dar, welche Hypothesen akzeptiert und welche verworfen werden (siehe Absatz 5.6).

Tabelle 5.41: Überprüfung der phasenbasierten Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C

Abhängige Variable	Zutreffende Hypothesen	Resultate
Total number of fixations	Ho	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht signifikant zwischen den Phasen.

	H0	Bei der phasenbasierten Betrachtung der total number of fixations zeigen sich keine signifikanten Unterschiede zwischen Experten und Novizen. Die Werte für diese Variable fallen für Experten etwas niedriger aus.
Average number of fixations	H0	Die total average of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C nicht signifikant zwischen den Phasen.
	H0	Bei der phasenbasierten Betrachtung der average number of fixations zeigen sich keine signifikanten Unterschiede zwischen Experten und Novizen. Die Werte für diese Variable fallen für Experten etwas niedriger aus.
Total fixation duration	H1	Für die total fixation duration können bei einem Review in der Programmiersprache C signifikante Unterschiede zwischen den Phasen festgestellt werden. Diese finden sich für Experten und Novizen, treten aber letzteren deutlicher hervor.
	H0	Hinsichtlich erfahrungsbasierter Unterschiede machen sich zwischen den Experten und Novizen in den einzelnen Phasen keine deutlichen Unterschiede bemerkbar. Die total fixation duration nimmt für beide Gruppen ähnliche Werte an.
Average fixation duration	H0	Die phasenbasierte Betrachtung der average fixation duration zeigt deutliche Unterschiede zwischen den einzelnen Phasen auf. Für Experten tritt ein signifikanter Unterschied zwischen Phase 2 und 3 hervor, für Novizen zwischen Phase 1 und 3.
	H1	Es treten bei der phasenbasierten Betrachtung der average fixation duration während eines Reviews in der Programmiersprache erfahrungsbedingte Unterschiede auf. Die Variable nimmt für Experten einen deutlich höheren Wert ans, als für Novizen.

Fixation rate	H1	Bei der phasenbasierten Betrachtung der fixation rate während eines Reviews in der Programmiersprache C zeigen sich signifikante Unterschiede zwischen den einzelnen Phasen. Für Experten zeigt sich ein signifikanter Unterschied zwischen Phase 1 und 3. Bei der Gruppe der Novizen machen sich zwischen allen Phasen Unterschiede bemerkbar.
	H0	Die Werte für die phasenbasierte Analyse der fixatin rate verhalten sich für beide Gruppen relativ identisch.
Total number of saccades	H0	Die phasenbasierte Betrachtung der total number of saccades bei einem Review in der Programmiersprache C zeigt keine Unterschiede zwischen den Phasen auf.
	H1	Die Werte für die die phasenbasierte Betrachtung der total number of saccades fallen für Experten deutlich niedriger aus, als für Novizen.
Average number of saccades	H0	Die phasenbasierte Betrachtung der average number of saccades bei einem Review in der Programmiersprache C zeigt keine Unterschiede zwischen den Phasen auf.
	H1	Die Werte für die die phasenbasierte Betrachtung der average number of saccades fallen für Experten deutlich niedriger aus, als für Novizen.
Total number of visits on erroneous lines	H1	Die phasenbasierte Betrachtung der total number of visits on erroneous lines zeigt mehrere Unterschiede zwischen den einzelnen Phasen auf. Sowohl für die Experten, als auch für die Novizen unterscheidet sich Phase 1 signifikant von Phase 2 und 3 und erzielt im Vergleich deutlich niederigere Werte.
	H1	Bei der phasenbasierten Betrachtung der total number of visits on erroneous lines zeigen sich für die Experten über alle Phasen hinweg niedrigere Werte als für die Novizen.

Average number of visits on erroneous lines	H1	Die phasenbasierte Betrachtung der average number of visits on erroneous lines zeigt mehrere Unterschiede zwischen den einzelnen Phasen auf. Sowohl für die Experten, als auch für die Novizen unterscheidet sich Phase 1 signifikant von Phase 2 und 3 und erzielt im Vergleich deutlich niederigere Werte.
	H1	Bei der phasenbasierten Betrachtung der average number of visits on erroneous lines zeigen sich für die Experten über alle Phasen hinweg niedrigere Werte als für die Novizen.
Total dwell time on erroneous lines	H1	Bezüglich der phasenbasierten Betrachtung der total dwell time on erroneous lines ergeben sich bei einem Code Review in der Programmiersprache C lediglich marginale Unterschiede zwischen den Phasen. Diese finden sich bei der Gruppe der Novizen zwischen der ersten und dritten Phase.
	H1	Im Vergleich zu den Novizen weisen die Experten in allen Phasen geringere Werte für die total dwell time on erroneous lines auf.
Average dwell time on erroneous lines	H1	Bezüglich der phasenbasierten Betrachtung der average dwell time on erroneous lines ergeben sich bei einem Code Review in der Programmiersprache C lediglich marginale Unterschiede zwischen den Phasen. Diese finden sich bei der Gruppe der Novizen zwischen der ersten und dritten Phase.
	H1	Im Vergleich zu den Novizen weisen die Experten in allen Phasen geringere Werte für die average dwell time on erroneous lines auf.

5.7 Zwischenfazit

Die Ergebnisse der hier vorgestellten Studie zeigen, dass bei Code Reviews in der Programmiersprache C erfahrungsbedingte Unterschiede zwischen Experten und Novizen auftreten und deren Durchführung in Phasen abläuft. Diese Phasen sind jeweils durch eine dominante Strategien gekennzeichnet.

Die deskriptive Statistik zu den grundlegenden Eye-Tracking-Metriken zeigt, dass

die Experten übergreifend eine effizientere Performance als die Novizen zeigen, jedoch sind die Unterschiede lediglich marginal und erreichen in den meisten Fällen kein statistisch signifikantes Niveau. Dennoch zeichnet sich dieser Trend in den Daten ab.

Was die phasenbasierte Betrachtung der Eye-Tracking-Daten anbelangt, so werden die erwarteten Unterschiede zwischen den Experten und Novizen bei diesen deutlicher und es zeigen sich signifikante Effekte. Insgesamt verhalten sich die untersuchten Metriken ähnlich zu den Vorhersagen, welche auf Basis der *holistic models of image perception* zu erwarten sind. Ebenso zeigen sich auch die in diesen Modellen beschriebenen Anteile von *global* und *focal viewing*, welche in ihrer Form und Funktion auch mit den bei Code Reviews beschriebenen *scan pattern* vergleichbar sind (Begel & Vrzakova, 2018; Busjahn et al., 2015; Kundel et al., 2007; Nodine & Kundel, 1987; Sharif et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980; Uwano et al., 2006). Die durch die phasenbasierte Betrachtung erlangten Ergebnisse legen auch nahe, dass die angesammelte Erfahrung, sei diese allgemein oder professionell, einen Einfluss auf die Durchführung der Reviews hat. Dies legt nahe, dass im Verlauf des Expertiseerwerbs Anpassungs- und Optimierungsprozesse stattfinden und sich Programmiererinnen und Programmierer an die Gegebenheiten ihrer Domäne anpassen (Ericsson & Towne, 2010; Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Gruber, 2007; Kok, 2016).

Die Ergebnisse, vor allem die deskriptive zu den untersuchten Eye-Tracking-Metriken, stellen sich zusammenfassend nicht so klar dar, wie auf Grundlage der verwendeten Theorien erhofft wurde. Zwei mögliche Ursachen könnten die relativ kleine und unausgewogene Stichprobengröße, sowie die rekursiven Prozesse der *holistic models of image perception* (Nodine & Kundel, 1987) sein. Letztere haben aus statistischer Sicht zur Folge, dass sich die Ausprägungen der Strategien durch einen kontrastiven Vergleich eher abschwächen und sich einem gemeinsamen Mittelwert annähern. Ähnliches zeigt sich auch in der Studie von Begel und Vrzakova (2018), die ebenfalls belegen können, dass sich bestimmte Betrachtungsstrategien während eines Code Reviews abwechseln und sich im zeitlichen Verlauf wiederholen.

Zusammenfassend kann über die C-Studie ausgesagt werden, dass sie belegt, dass es bei der Durchführung eines Code Reviews zu einer phasenbasierten Vorgehensweise kommt, welche Ähnlichkeiten zu den *holistic models of image perception* aufweist (Kundel et al., 2007; Nodine & Kundel, 1987; Sheridan & Reingold, 2017; Swensson, 1980). Auch vergleichbare Studien (Begel & Vrzakova, 2018; Busjahn et al., 2015; Sharif et al., 2012; Peachock et al., 2017; Uwano et al., 2006) deuten Ähnlichkeiten an. Die Einbeziehung der Expertiseforschung liefert darüber hinaus auch Hinweise darauf, dass sich vor allem die *professionelle Programmiererfahrung* signifikant auf die Leistung während eines Code Reviews auswirkt und ein Zusammenhang zu den gezeigten Augenbewegungen besteht (Ericsson & Towne, 2010; Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023; Kok, 2016). Ergänzend dazu sollte für zukünftige

Studien in diesem Bereich überlegt werden, ob weitere Metriken (z.B. *linearity, transitions, ...*), beispielsweise aus der Leseforschung einbezogen werden (Busjahn et al., 2015; Obaidellah et al., 2018; Sharafi et al., 2015; Strukelj & Niehorster, 2018) . Diese könnten beispielsweise die Blickbewegungen während des *focal viewings* deutlicher beschreiben und ließen sich in die *holistic models of image perception* integrieren.

Kapitel 6

Studie 2: Visuelle Expertise bei Code Reviews in der objektorientierten Programmiersprache C++

Dieses Kapitel basiert auf dem Konferenzbeitrag *Code Reviews in C++: Preliminary Results from an Eye-Tracking Study* (Hauser, Schreistetter et al., 2020), welcher im Rahmen des von ACM organisierten *Eye Movements in Programming* 2020 in Stuttgart vorgestellt wurde. Auf Grundlage der in dieser Studie vorgestellten Daten wurde eine weitere Studie erstellt, welche die Anwendung der *holistic models of image perception* in Bezug auf die Datenauswertung darstellt. Diese wurde zum Zeitpunkt der Veröffentlichung dieser Dissertation ebenfalls für den Workshop *Eye Movements in Programming* eingereicht, welcher im Juni 2024 in Glasgow stattfinden soll.

Kurzfassung

Diese Studie beschäftigt sich mit der Erforschung von visueller Expertise bei der objektorientierten Programmiersprache C++. Sie kann auf die Daten von 34 Versuchspersonen zurückgreifen, welche sich in 18 Novizen und 16 Experten unterteilen. Als Teil des Experiments begutachten diese insgesamt acht Codebeispiele. Sechs dieser Beispiele sind fehlerhaft, wobei zwei der Quellcodes mehrere Fehler beinhalten. Während des Reviews werden die Augenbewegungen von einem Tobii Pro Spectrum 600 aufgezeichnet. Ergänzend dazu werden demografische Daten der Versuchspersonen erhoben und ein stimulusbasiertes Interview zur Verifikation der erhobenen Eye-Tracking-Daten durchgeführt. Die Datenauswertung analysiert sowohl die grundlegenden Eye-Tracking-Metriken, als auch phasenbasierte Veränderungen vor dem Hintergrund von visueller Expertise. Bezüglich der Interpretation wird auf die *holistic models of image perception* zurückgegriffen. Es zeigt sich, dass die holistic models of image perception für die Anwendung auf C++-Quellcode geeignet sind. Zusätzlich

kann beobachtet werden, dass die Code Reviews in Phasen ablaufen, welche durch bestimmte Strategien (z.B. Scan, Fehlersuche, ...) gekennzeichnet sind. Ergänzend dazu zeigen sich zwischen Experten und Novizen erfahrungsbedingte Unterschiede, welche unterstreichen, dass Experten elaborierte Strategien einsetzen und über eine im Vergleich zu den Novizen bessere Informationsaufnahme und -verarbeitung bei Quellcode in der Programmiersprache C++ verfügen.

6.1 Design

Die nachfolgenden Absätze geben einen Überblick darüber, auf welchem Forschungsdesign diese Studie basiert (siehe Absatz 6.1.1) und welche Variablen bzw. Metriken untersucht werden (siehe Absatz 6.1.2).

6.1.1 Forschungsdesign

Die Studie greift auf das bereits im vorherigen Kapitel vorgestellte Design zurück, welches auf den Studien von Uwano eta al. (2006) und Sharif et al. (2012) basiert und leichten Modifikationen unterzogen wird. Diesbezüglich wird erneut ein kontrastiver Vergleich zwischen Novizen und Experten eingesetzt, in dessen Rahmen die beiden Erfahrungsstufen miteinander verglichen und gegenübergestellt werden. Ergänzend dazu werden die Annahmen, der verschiedenen holistischen Modelle der Bildbetrachtung aufgegriffen (Kundel et al., 2007; Nodine & Kundel, 1987; Sheridan & Reingold, 2017; Swensson, 1980), und untersucht, inwiefern ein pahsenbasiertes Vorgehen während des Reviews beobachtet werden kann. Die resultierenden Phasen werden sowohl innerhalb der Gruppen, als auch übergreifend betrachtet und auf mögliche Unterschiede hin verglichen (siehe Absatz 4.3 in Kapitel 4).

6.1.2 Relevante Variablen und Metriken

Die abhängigen Variablen dieser Studie basieren auf den von Sheridan und Reingold (2017,S.5) vorgestellten und in Tabelle 2.1 genannten gemeinsamen Metriken der verschiedenen *holistic models of image perception*. Der Fokus der hier durchgeführten Studie liegt schwerpunktmäßig auf der Aufnahme und Verarbeitung von Informationen aus den vorgestellten Codebeispielen, weshalb den fixationsbasierten Metriken eine besondere Bedeutung im Studiendesign zuteil wird. Diese werden noch um die *Anzahl der erzielten Punkte* bzw. *Anzahl der gefundenen Fehler* ergänzt, welche aussagt, wie erfolgreich eine Versuchsperson bei der Beurteilung der Codebeispiele abgeschnitten hat.

Konkret als abhängige Variablen genutzt werden dabei:

- *Anzahl der gefundenen Fehler*
- *Total number of fixations*
- *Average number of fixations*

- *Total fixation duration*
- *Average fixation duration*
- *Fixation rate*
- *Total number of saccades*
- *Average number of saccades*
- *Total number of visits on erroneous lines*
- *Average number of visits on erroneous lines*
- *Total dwell time on erroneous lines*
- *Average dwell time on erroneous lines*

Als unabhängige Variablen fungieren in dieser Studie erneut die *allgemeine Programmiererfahrung* und die *berufliche Programmiererfahrung* der Versuchspersonen.

6.2 Methoden

Nachfolgend wird die Methodologie dieser Studie beschrieben. Im Absatz 6.2.1 wird auf die verwendeten Instrumente eingegangen. Darauf folgend wird die Stichprobe (siehe Absatz 6.2.2) beschrieben, bevor abschließend im Absatz 6.2.3 die Durchführung des Experiments geschildert wird.

6.2.1 Instrumente

Für die Durchführung des Experiments wurden vier Instrumente benötigt. Dabei handelt es sich um die Codebeispiele, die zu untersuchenden Codebeispiele, einen Eye-Tracker, einen qualitativen Fragebogen und einen Interviewleitfaden. Diese werden nachfolgend näher beschrieben.

Codebeispiele als visuelle Stimuli

Wie bereits zuvor angemerkt wurde, wurden im Rahmen dieser Studie Codebeispiele aus einer objektorientierten Programmiersprache eingesetzt. Im Gegensatz zum Großteil der vergleichbaren Studien, die sich mit diesem Bereich befassen (siehe Kapitel 3), wurde im hier präsentierten Fall bewusst nicht die Sprache Java genutzt (Obaidellah et al., 2018; Sharafi et al., 2015). Stattdessen entschied man sich für die Programmiersprache C++. Diese erfreut sich großer Beliebtheit und wird zur Realisierung von Softwareprojekten genutzt. Aufgrund ihrer hohen Verbreitung in der Industrie, Wirtschaft und Hochschullehre, konnte für die Rekrutierung der Stichprobe ein größerer Adressatenkreis angesprochen werden (TIOBE, 2022).

Insgesamt wurden für die Datenerhebung acht kurze Codebeispiele generiert. Diese wurden von Mitarbeitern des LaS^3 im Rahmen eines Wettbewerbs erstellt und

anschließend einer Begutachtung durch Experten unterzogen. Bezüglich der Anforderungen an die Beispiele galt es zu beachten, dass diese aufgrund der Scroll-Problematik bei modernen Eye-Trackern (Tobii Pro, 2020, 2021) nicht zu lang sein durften (max. 50 Codezeilen), so dass diese vollständig auf einem normalen Monitor (24″) angezeigt werden können. Weiterhin mussten die verwendeten Codes sowohl für Novizen, als auch für Experten verständlich und lösbar sein, dabei jedoch auch bis zu einem gewissen Grad herausfordernd sein. Im Vergleich zur direkten Vorgängerstudie (siehe Kapitel 5) wurden in diesem Fall auch korrekte Beispiele erstellt, welche auf der einen Seite als Distraktoren dienen, gleichzeitig aber auch Erkenntnisse über das Verhalten der Versuchspersonen beim Nichtvorhandensein eines Fehlers erbringen sollen. Bezüglich der enthaltenen Fehlerarten handelt es sich wie in der Vorgängerarbeit erneut um logische Fehler. Diese sind nicht auf den ersten Blick ersichtlich und bedürfen eines tieferen Verständnisses des Codes.

Alle acht Codebeispiele verzichten auf Highlightings (Beelders & du Plessis, 2016; Moser, 2022; Schorr, o. J.; Peterson et al., 2019). Es handelt sich lediglich um schwarzen Code in der Schriftgröße 14 auf einem weißen Hintergrund. Als Schriftart wurde bei der Erstellung Courier New verwendet. Diese Darstellungsform stellt sich für Entwickler als unüblich dar, da in den meisten Situationen Darstellungsoptionen (z.B. Syntax- oder Semantic-Highlighting) mit entsprechender Farbunterlegung bevorzugt werden (Schorr, o. J.). Durch die neutrale Farbgebung sollen eventuelle Vor- oder Nachteile, die durch das Highlighting entstehen können möglichst minimiert und für alle Probanden gleiche Bedingungen geschaffen werden (Beelders & du Plessis, 2016; Peterson et al., 2019).

Die acht ausgewählten Codebeispiele decken eine breite Variation der in C++ möglichen logischen Fehler ab und sollen nachfolgend kurz thematisiert werden:

Stack error (LOC=12): Ein *stack error* ist ein Laufzeitfehler, der dann auftritt, wenn einem Programm der Speicher im Aufrufstapel ausgeht. Dabei handelt es sich um einen logischen Fehler. Allgemein betrachtet deutet ein *stack error* auf ein Problem bei der Ressourcenbereitstellung hin. Damit das Programm ausgeführt und der Speicher ordnungsgemäß genutzt werden kann, muss dieser Fehler behoben werden (Computer Hope, 2017; Techopedia, 2022; Thornton, 2016).

```
#include <iostream>

std::string& Concat(std::string a, std::string b) {
        std::string s = '''';
        s.append(a);
        s.append(b);
        return s;
}
int main() {
        std::string& s = Concat(''string1'', ''string2'');
        std::cout << s.c_str();
}
```

Listing 6.1: Codebeispiel zum *stack error*)

Non-virtual destructor (LOC=31): Im Falle dieses Beispiels wird fälschlicherweise ein sogenannter *non-virtual destructor* verwendet. Dieser sollte eigentlich as *virtual destructor* verwendet werden. Bei Ausführung wird in diesem Beispiel das entsprechende Objekt beschädigt und ein nicht vorhersehbares bzw. nicht lauffähiges Resultat zur Folge haben (GeeksforGeeks, 2021; Intel Corporation, 2010).

```
class Account
{
public:
        Account(std::string name)
        {
                this->name = new std::string(name);
        }

        std::string *name;

        ~Account()
        {
                delete name;
        }
};

class MyAccount : public Account
{
public:
        MyAccount(std::string name, std::string address) :Account(name)
        {
                this->address = new std::string(address);
        }

        std::string *address;

        ~MyAccount()
        {
                delete address;
        }
};

int main()
{
        Account *account = new MyAccount(''MyName'', ''MyAddress'');
        delete account;
}
```

Listing 6.2: Codebeispiel zum *non-virtual destructor*

Missing method override (LOC=29): Das Keyword *override* wird bei C++ verwendet um die virtuelle Methode einer Basisklasse außer Kraft zu setzten. In diesem Codebeispiel wurde es versäumt, diesen Prozess zu implementieren, daher wird die entsprechende Klasse abstrakt und verursacht einen Fehler (ISO C++, 2022; Pencil Programmer, 2022).

```
#include <iostream>

class Square
{
public:
```

```
        Square(float a)
        {
                this->a = a;
        }

        float a;

        virtual float Volume()
        {
                return a * a;
        }
};

class Rectangle : public Square
{
public:
        Rectangle(float a, float b) :Square(a)
        {
                this->b = b;
        }
private:
        float b;
};

int main()
{
        Rectangle *rectangle = new Rectangle(2, 5);
        std::cout << rectangle->Volume();
}
```

Listing 6.3: Codebeispiel zum *missing method override*

Slicing problem (LOC=28): Das sogenannte *slicing problem* tritt bei C++ (wie auch in diesem Codebeispiel dargestellt) dann auf, wenn ein Objekt eines Unterklassentyps in ein Objekt eines Oberklassentyps kopiert wird. Die Kopie der Oberklasse hat dann keine der in der Unterklasse definierten Mitgliedsvariablen. Diese werden sozusagen "abgeschnitten". Weiterhin kann das *slicing problem* auch auftreten, wenn beispielsweise ein Objekt eines Unterklassentyps durch den Zuweisungsoperater der Oberklasse in ein Objekt desselben Typs kopiert wird. In diesem Fall behalten einige der Mitgliedsvariablen des Zielobjekts ihre ursprünglichen Werte, anstatt vom Quellobjekt übernommen zu werden. Das *slicing problem* ist nicht nur auf C++ limitiert, es kann auch in anderen objektorientierten Programmiersprachen angetroffen werden. Aufgrund seiner programmiersprachenspezifischen Charakteristiken ist es jedoch in C++ am häufigsten anzutreffen (Geeks for Geeks, 2017, 2022; Learn C++, 2022).

```
#include <iostream>

class Account
{
public:
        std::string nickname = ''nick'';

        virtual std::string GetName() {
                return nickname;
        }
```

```
};

class AccountWithRealName : public Account
{
public:
        std::string firstname = ''first'';
        std::string lastname = ''last'';

        virtual std::string GetName() {
                return nickname.append(firstname).append(lastname);
        }
};

void ProcessAccount(Account a)
{
        std::cout << a.GetName().c_str();
}

int main()
{
        AccountWithRealName a;
        ProcessAccount(a);
}
```

Listing 6.4: Codebeispiel zum *slicing problem*

Korrektes Codebeispiel 1 (LOC=28): Das Beispiel nutzt veraltete Styleguides und könnte somit aus formaler Sicht verbessert werden, weist aber keinen logischen Fehler auf und ist lauffähig.

```
#include <iostream>
using namespace std;

class Adder {
   public:
      // constructor
      Adder(int i = 0) {
         total = i;
      }

      // interface to outside world
      void addNum(int number) {
         total += number;
      }

      // interface to outside world
      int getTotal() {
         return total;
      };

   private:
      // hidden data from outside world
      int total;
};

int main() {
   Adder a(0);

   a.addNum(10);
```

```
    a.addNum(20);
    a.addNum(30);

    cout << ''Total '' << a.getTotal() <<endl;
    return 0;
}
```

Listing 6.5: Codebeipiel für einen lauffähigen Quellcode

Size of dynamic array (LOC=40): Dieses Codebeispiel weist insgesamt drei Fehler auf und stellt sich somit als komplexer dar, als die bisher gezeigten Stimuli (Kononenko et al., 2016; Mäntylä & Lassenius, 2009). Beim ersten Fehler gibt der Befehl *sizeof* lediglich die Größe des Zeigers auf das erste Element des Arrays zurück, nicht jedoch die tatsächlich Größe des Arrays. Zusätzlich dazu beinhaltet das Beispiel noch zwei sogenannte *memory leaks* (Boyini, 2019; Dieterich, 2009; Dong & Yang, 2019; ISO C++, 2022; Rai, 2019).

```
#include <cstdint>
#include <string>

class Material{
private:
    uint64_t id;
    std::string *bezeichnung;
public:
    Material(uint64_t id, std::string bezeichnung){
        this->id= id;
        this->bezeichnung= new std::string(bezeichnung);
    }
};

class Regal{
private:
    std::string *standort;
    Material **inhalt;
public:
    Regal(std::string standort, uint32_t groesse){
        this->standort= new std::string(standort);
        inhalt= new Material*[groesse];
    }
    ~Regal(){
        delete[] inhalt;
    }
    void einlagern(Material *mat){
        uint32_t groesse= (sizeof(inhalt)/sizeof(Material));
        for(uint32_t i = 0; i < groesse; i++){
            if (inhalt[i] == NULL) {
                inhalt[i]= mat;
            }
        }
    }
};

int main() {
        Regal regal= Regal(''Raum A, Gebaeude B'', 5);
    Material schrauben= Material(0,''Schrauben'');
    regal.einlagern(&schrauben);
    return 0;
```

```
}
```

Listing 6.6: Codebeispiel zum *size of dynamic array*

Missing copy constructor (LOC=40): Auch dieses Codebeispiel enthält drei Fehler und soll somit einen höheren Komplexitätsgrad erreichen, als die Quellcodes 1-4 (Kononenko et al., 2016; Mäntylä & Lassenius, 2009; Thomas, 2010). Bei einem der vorhandenen Fehler handelt es sich um einen *missing copy constructor*. Dieser Fehler tritt dann auf, wenn Objekte einer bestimmten Klasse kopiert werden sollen, aber die Klasse über keinen *copy constructor* verfügt (Dieterich, 2009; Dong & Yang, 2019; Geeks for Geeks, 2024). Ebenso treten erneut zwei *memory leaks* auf (Boyini, 2019; ISO C++, 2022; Rai, 2019).

```
#include <cstdint>
#include <string>

class Motor
{
private:
    uint16_t ps;
    uint8_t zylinderanzahl;
public:
    Motor(uint16_t ps, uint8_t zylinderanzahl){
        this->ps= ps;
        this->zylinderanzahl= zylinderanzahl;
    }
    void setPS(uint16_t ps){
        this->ps= ps;
    }
};

class Auto{
private:
    Motor* motor;
    std::string *kennzeichen;
public:
    Auto(std::string kennzeichen, Motor* motor){
        this->kennzeichen= new std::string(kennzeichen);
        this->motor= motor;
    }
    void setKennzeichen(std::string kennzeichen){
        this->kennzeichen= new std::string(kennzeichen);
    }
    Motor* getMotor(){
        return motor;
    }
};

int main() {
    Motor motor= Motor(90,3);
        Auto flitzer= Auto(''R CY-55'',&motor);
    Auto tuningFlitzer= flitzer;
    tuningFlitzer.setKennzeichen(''R M-99'');
    tuningFlitzer.getMotor()->setPS(200);
    return 0;
}
```

Listing 6.7: Codebeispiel zum *missing copy constructor*

Korrektes Codebeispiel 2 (LOC=28): Auch dieses Beispiel weist Verstöße gegen Styleguides auf, wäre ist lauffähig.

```
#include <iostream>

class A
{
public:
        void print()
        {
                std::cout << ''A'';
        }
};

class B
{
public:
        void print()
        {
                std::cout << ''B'';
        }
};

class C : public A, public B
{
public:
        void print() {
                std::cout << ''C'';
        }
};

int main() {
        C c;
        c.print();
}
```

Listing 6.8: Codebeipiel für einen lauffähigen Quellcode

Verwendeter Eye-Tracker

Eye-Tracking-Studien im Bereich der Code Reviews stellen hohe Anforderungen an die eingesetzte Hardware. Im Vergleich zur Vorgängerstudie (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023), wurden bei dieser Arbeit längere Codebeispiele genutzt, wodurch sich die Anforderungen an die Genauigkeit des Eye-Trackers noch etwas weiter gesteigert haben. Durch die zu erwartende längere Bearbeitungszeit musste unter anderem bedacht werden, dass der beim Eye-Tracking auftretende *Drift* der Augen bestmöglich kompensiert werden kann (Duchowski, 2017; Holmqvist et al., 2011; Nyström, Niehorster, Andersson & Hooge, 2021). Die Minimalanforderung lautete erneut eine Genauigkeit von <.400° und eine Aufnahmefrequenz von mindestens 120Hz. Im Falle dieser Studie entschied man sich dazu, auf Hardware aus dem Regensburger Eye-Tracking-Classroom zurückzugreifen.

Für die Datenerhebungen wurden folglich drei Tobii Spectrum mit einer Aufnahmefrequenz von 600Hz genutzt. Im Laborbetrieb und bei bestmöglicher Kalibrierung, können diese Geräte eine Genauigkeit von .100° erreichen und für die Dauer eines Experiments aufrechterhalten (Tobii Pro, 2020). Die drei Eye-Tracker wurden im Rahmen der Studie parallel betrieben. Jeder der Eye-Tracker wurde von einer entsprechend geschulten Person bedient. Im Anschluss an die Datenerhebungen wurden die erhobenen Daten zu einem gemeinsamen Datensatz kombiniert.

Qualitativer Fragebogen

Im Sinne einer methodischen Triangulation wurde ein qualitativer Fragebogen entworfen und den Versuchspersonen im Rahmen der Studie vorgelegt. Dieser beinhaltete eine DSGVO-konforme Einverständniserklärung, welcher die Versuchspersonen vor dem Beginn des Experiments zustimmen mussten. In dieser wurde den Forschenden die Erlaubnis erteilt, persönliche Daten und Augenbewegungen zu erheben. Dies schloss unter anderem die Aufzeichnung, Bearbeitung, Auswertung und Erstellung von wissenschaftlichen Arbeiten ein.
Bei den abgefragten persönlichen Daten handelte es sich um das Alter, das Geschlecht und den Beruf. Ergänzend dazu wurden programmierspezifische Daten abgefragt. So sollten die Versuchspersonen angeben, welche Erfahrungen sie als Programmierer haben (z.B. Zusatzausbildungen, spezielle Kenntnisse, ...), seit wie vielen Jahren sie programmieren und über wie viele Jahre professionelle Programmiererfahrung sie verfügen. Weiterhin wurden die Versuchsteilnehmerinnen und -teilnehmer um eine kurze Selbsteinschätzung ihrer Fähigkeiten gebeten. Mittels einer vierstufigen Likert-Skala sollte angegeben werden, ob man die eigenen Programmierkenntnisse als gut einschätzt, ob man sich selbst in der Programmiersprache C++ als gut einschätzt und ob man davon ausgeht, ein guter Reviewer zu sein.
In einem letzten Abschnitt konnten die Versuchspersonen auf freiwilliger Basis angeben, ob sie auch zukünftig an Studien teilnehmen möchten. Falls sie zustimmten, konnten sie ihre Kontaktdaten angeben und sich im Falle von neuen Projekten benachrichtigen lassen.

Interviewleitfaden

Ergänzend zum qualitativen Fragebogen und wie bereits in der Vorgängerstudie, enthielt auch diese Datenerhebung ein stimulusbasiertes Interview, in welchem den Versuchspersonen die eigenen Augenbewegungen präsentiert wurden. Zur Durchführung der Interviews wurde ein offener Interviewleitfaden erstellt. Dieser enthielt vorab formulierte Leitfragen, die bei jedem Interview gestellt werden mussten. Diese lauteten folgendermaßen:

- "Wie haben Sie die Datenerhebung empfunden?"
- "Haben Sie die Codebeispiele eher als schwierig oder als leicht erachtet?"

- "Haben Sie bei der Durchführung eine bestimmte Vorgehensweise angewandt?"

Bei besonderen Auffälligkeiten (z.B. sehr lange *fixation duration* auf ein bestimmtes Areal, häufige Transitionen zwischen Codesegmenten) sah das Konzept des stimulsbasierten Interviews vor, dass diese gezielt hinterfragt wurden. Typische Fragen waren diesbezüglich zum Beispiel:

- "Warum haben Sie sich so sehr auf diese Stelle konzentriert? Waren dort für Sie wichtige Informationen?"

- "Weshalb haben sie mit der Durchsicht des Codes bei der *main* begonnen? Hatte das einen bestimmten Grund oder Vorteil für Sie?"

Insgesamt verfolgten die stimulusbasierten Interviews zwei zentrale Ziele:

1. Die aufgezeichneten Augenbewegungen sollten einer Validierung unterzogen werden. Es sollte sichergestellt werden, dass mögliche Störeinflüsse (z.B. Ablenkung der Versuchsperson, technische Probleme bei der Kalibrierung der Eye-Tracker, Überforderung, ...) identifiziert und für die weitere Datenauswertung berücksichtigt werden konnten. Gleichzeitig sollte sichergestellt werden, dass sich die Versuchspersonen über ihre eigenen Augenbewegungen im Klaren waren und diese (zumindest zum Teil) erklären konnten.

2. Die stimulsbasierten Interviews sollten Rückschlüsse über die von den Versuchspersonen angewandten kognitiven Strategien geben. Dies diente vorrangig dazu, bestimmte Vorgehensweisen von Experten und Novizen näher beleuchten zu können und Unterschiede zwischen den Erfahrungsstufen zu erkennen.

6.2.2 Stichprobe

Im Vergleich zur Vorgängerarbeit konnte für diese Studie eine größere Stichprobe von 40 Versuchspersonen rekrutiert werden. Nach Sichtung und Bereinigung der Datensätze mussten sechs Versuchspersonen aufgrund von Kalibrierungsproblemen oder Datenlücken ausgeschlossen werden. Somit standen zur Analyse die Daten von insgesamt 34 Probanden zur Verfügung. Die höhere Anzahl an Probanden geht auf eine Reihe von Veränderungen im Erhebungsverfahren zurück. Konkret handelt es sich dabei um vier Punkte:

1. *Eye-Tracking-Classroom:* Diese Studie konnte erstmals auf die Möglichkeiten des Regensburger Eye-Tracking-Classroom zurückgreifen, wodurch parallele Datenerhebungen an mehreren Probanden zur gleichen Zeit möglich wurden. Somit konnten die zur Verfügung stehenden Zeitslots besser genutzt werden.

2. *Anwerbung von Probanden aus Netzwerken:* Die im Rahmen der Vorgängerstudien etablierten Netzwerke zur Rekrutierung von Probanden konnten bei dieser Studie erneut genutzt werden. Weiterhin wurden im Voraus auch Kontakte zu Experten-Netzwerken im bayerischen Raum hergestellt und zur Anwerbung von erfahrenen Probanden genutzt.

3. *Verlängerte Datenerhebung:* Der für die Datenerhebung geplante Zeitraum für diese Studie wurde im Vergleich zur Vorgängerstudie ebenfalls deutlich erweitert. Insgesamt beläuft sich dieser im Falle des vorliegenden Experiments auf fast sieben Monate.

4. *Gesteigerte Mobilität:* Im Zuge des Aufbaus des Regensburger Eye-Tracking-Classroom wurde auch Equipment für mobile Datenerhebungen beschafft. Obwohl es sich beim Tobii Pro Spectrum schon alleine aufgrund seiner Größe und seines Gewichts (Tobii Pro, 2020) nicht um einen Eye-Tracker handelt, der auf Mobilität ausgelegt ist, standen drei Geräte mit entsprechend ausgestatteten Laptops für Feldstudien zur Verfügung. Diese wurden vorwiegend für Datenerhebungen bei Probanden, insbesondere bei Expertinnen und Experten eingesetzt, welche meist nur unter großem Aufwand rekrutiert werden konnten. In diesem Fall galt das Motto: *"Bring den Eye-Tracker zum Probanden, nicht den Probanden zum Eye-Tracker"*.

Die Stichprobe setzt sich aus 30 Männern und vier Frauen zusammen. Eine ähnliche geschlechtsspezifische Verteilung zeigte sich auch in den Vorgängerstudien, die in Regensburg durchgeführt worden sind (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023; Hutzler et al., 2018; Nivala et al., 2016). Geschlechtsspezifische Unterschiede bei Code Reviews sind zwar nicht Gegenstand der hier dargestellten Studie und werden in dieser auch nicht weiter diskutiert, rücken aber zunehmend mehr in den Forschungsfokus (Hou et al., 2013). Der aktuelle Forschungsstand zu genderspezifischen Unterschieden bei Code Reviews deutet darauf hin, dass diese in einem marginalen Rahmen auftreten können, jedoch deutlich mehr Forschung zu diesem Bereich notwendig ist, um zu gesicherten Erkenntnissen zu kommen (Obaidellah et al., 2018; Sharafi et al., 2015).
Bezüglich des Alters der Versuchspersonen wird eine Spanne von 17 Jahren abgedeckt, welche sich von 21 bis 38 Jahren erstreckt. Das allgemeine Durchschnittsalter wird mit einem Mittelwert von 25.12 (SD = 3.80) Jahren angegeben, wobei sich in der Stichprobe ein deutlicher Ausschlag um das zweiundzwanzigste Lebensjahr bemerkbar macht (siehe Grafik 5.3.1). Dies lässt sich vor allem darauf zurückführen, dass ein Großteil der Novizen aus Bachelorstudiengängen der Fakultät EI der OTH Regensburg rekrutiert worden ist.
Wie auch in der Vorgängerstudie (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023) wurde die Erfahrung der Versuchspersonen auf zwei Skalen gemessen. Dabei handelt es sich um *allgemeine Programmiererfahrung in Jahren* und *professionelle Programmiererfahrung in Jahren*. Unter der *allgemeinen Programmiererfah-*

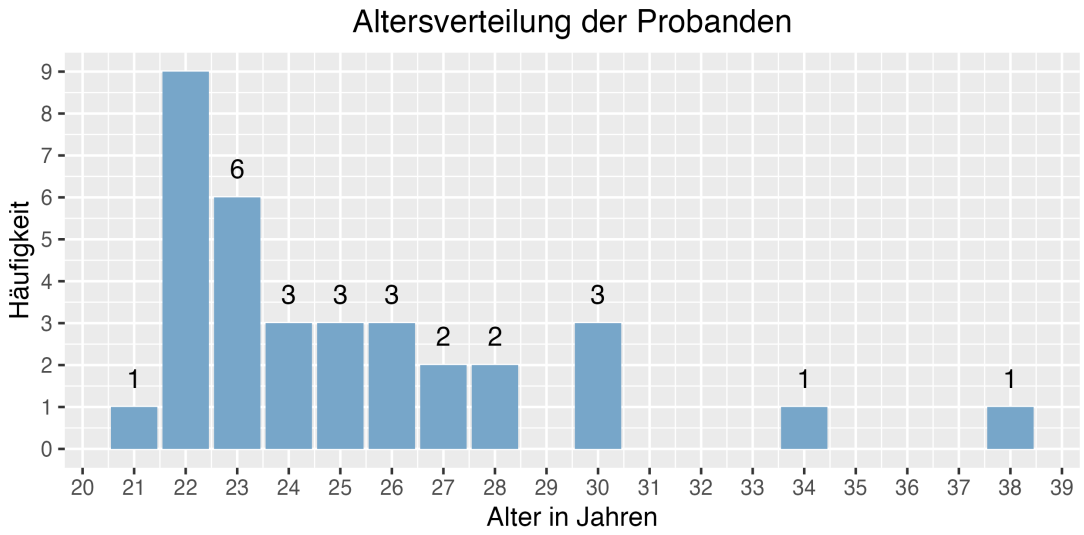

Abbildung 6.1: Altersverteilung der Probanden

rung ist der gesamte Zeitraum zu verstehen, in welchem sich die Versuchspersonen mit dem Programmieren befassen. *Professionelle Programmiererfahrung* meint hingegen, den Zeitraum, in welchem einer beruflichen Tätigkeit nachgegangen wird, welche ihren Fokus auf programmierspezifische Aufgaben richtet und mit welcher ein signifikanter Teil des monatlichen Einkommens generiert wird.
Für die *allgemeine Programmiererfahrung* ergibt sich ein Mittelwert von 5.88 (SD = 4.93) Jahren. Die Spannweite für diese Variable deckt 23 Jahre ab, wobei die Versuchspersonen mit der wenigsten Erfahrung auf zwei Jahre Programmiererfahrung zurückgreifen können. Umgekehrt lässt sich auf Seiten des Maximums eine Angabe von 25 Jahren finden.
Die *professionelle Programmiererfahrung* weist eine ähnliche Struktur auf. Im Durchschnitt beläuft sich diese in der Stichprobe auf 2.25 (SD = 3.25) Jahre und deckt eine Spannweite von 13 Jahren ab. Das Minimum stellen dabei null Jahre und das Maximum 13 Jahre dar.
Die Einteilung der Stichprobe in zwei Expertenstufen erfolgt auf Basis der allgemeinen Programmiererfahrung. Die ursprüngliche Planung, die Versuchspersonen in drei Expertenstufen á 20 Personen zu unterteilen, musste im Zuge der andauernden COVID-19-Pandemie und der mangelnden Vergleichbarkeit mit der Vorgängerstudie verworfen werden. Ziel hinter dieser Einteilung wäre es gewesen, die Zwischengruppe der fortgeschrittene Probanden genauer betrachten zu können und Rückschlüsse über deren Fähigkeiten zu Beginn einer beruflichen Tätigkeit gewinnen zu können. Die aktuelle zweistufige Unterteilung setzt sich folgendermaßen zusammen:

- *Novizen:* Versuchspersonen dieser Erfahrungsstufe haben weniger als fünf Jahre allgemeine Programmiererfahrung. Diese Charakteristik trifft auf insgesamt 18 Personen in der Stichprobe zu.

- *Fortgeschrittene/Experte:* Um als Fortgeschrittene bzw. Experte eingestuft zu werden, müssen die Versuchspersonen mindestens auf fünf Jahre allgemeine oder

mehr zurückgreifen. Sie sind mit 16 Probanden in der Stichprobe vertreten.

Die Verteilung der *allgemeinen* und *professionellen Programmiererfahrung* wird gruppenübergreifend in den beiden Abbildungen 6.2 und 6.3 dargestellt.

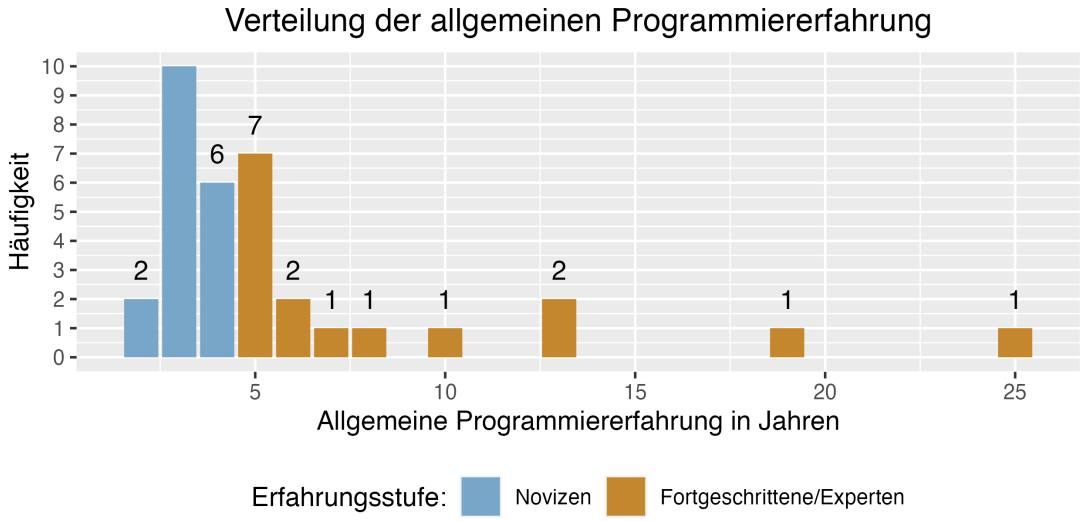

Abbildung 6.2: Verteilung der allgemeinen Programmiererfahrung

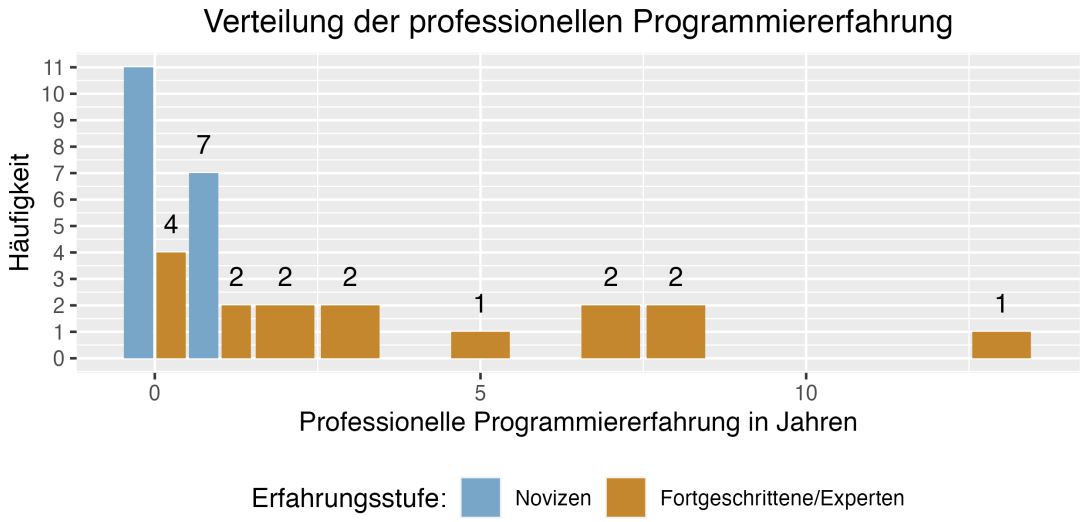

Abbildung 6.3: Verteilung der professionellen Programmiererfahrung

6.2.3 Durchführung

Die Rekrutierung der Stichprobe für dieses Experiment erfolgte mit ca. zwei Wochen Vorlauf zur Versuchsdurchführung. Zur Gewinnung von Novizinnen und Novizen wurde in Lehrveranstaltungen der Computerwissenschaften nach interessierten Personen gefragt. Sofern dies für diese Gruppe möglich war, konnten die Probanden die Teilnahme an der Eye-Tracking-Studie mit zu absolvierenden Pflichtübungen verrechnen. Wie auch in der Vorgängerstudie, gestaltete sich die Anwerbung von fortge-

schrittenen Programmiererinnen und Programmierern bzw. Expertinnen und Experten als schwieriger. Im Falle der vorliegenden Studie wurde auf bereits eingerichtete Netzwerke zurückgegriffen, welche es ermöglichten Versuchspersonen aus der Wirtschaft und Industrie zu rekrutieren. Weiterhin konnten auch Doktorandinnen und Doktoranden aus den Computerwissenschaften gewonnen werden.
Die Datenerhebungen zu dieser Studie fanden zum Großteil im Regensburger Eye-Tracking-Classroom statt. Dabei wurden drei Tobii Pro Spectrum parallel genutzt. Weiterhin konnten aufgrund des damit einhergehenden Laborsettings äußere Störeinflüsse auf ein Minimum reduziert und die Experimente in einer kontrollierten Umgebung durchgeführt werden. Im Falle der Expertinnen und Experten galt das Motto *"Bring das Labor zum Probanden, nicht den Probanden zum Labor"*. Diese Vorgehensweise war dadurch bedingt, dass die Expertinnen und Experten in den meisten Fällen zentrale Bestandteile ihrer jeweiligen Teams waren und in der regulären Arbeitszeit nicht oder nur sehr eingeschränkt verfügbar waren und somit eine Anreise in den Eye-Tracking-Classroom nicht möglich gewesen wäre. Dies bedingte, dass im Rahmen einer C++-User-Group-Veranstaltung in München mobile Datenerhebungen stattfanden. Dazu wurden erneut drei Tobii Pro Spectrum verwendet. Diese konnten in einer geeigneten Räumlichkeit aufgebaut und betrieben werden. Es gab vorab die Möglichkeit den zur Verfügung gestellten Raum zu inspizieren und störende Einflüsse zu identifizieren und zu minimieren.
Die Datenerhebung folgte dem Beispiel der im vorherigen Kapitel 5 beschriebenen Studie (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023) Sie unterteilte sich erneut in vier Phasen:

1. *Aufklärungsgespräch und Instruktion:* Die Versuchspersonen wurden vor der Datenerhebung über die Studie informiert und gaben ihr Einverständnis für die Teilnahme. Nach Einholung der Einverständniserklärung wurden die Teilnehmerinnen und Teilnehmer instruiert, was ihre Aufgaben im Experiment sind.

2. *Ausfüllen des Fragebogens:* Vor Beginn der Studie wurden die Versuchspersonen gebeten, den erwähnten Fragebogen auszufüllen. Dieser wurde ihnen in Papierform ausgehändigt und für die weitere Datenauswertung aufbewahrt.

3. *Erhebung der Eye-Tracking-Daten:* Das Eye-Tracking-Experiment selbst wurde in Tobii Pro Lab aufgesetzt und auch in diesem aufgezeichnet. Die Datenerhebung unterteilte sich dabei in vier Blöcke, von denen jeder jeweils zwei Code-Stimuli und nachgeschaltete Antwortmöglichkeiten enthielt. Zwischen den Blöcken hatten die Versuchspersonen die Möglichkeit, sich zu entspannen, Fragen zu stellen oder gegebenenfalls das Experiment zu beenden. Jeder Block begann mit einer Kalibrierung des Eye-Trackers, wodurch eventuell auftretende Ungenauigkeiten ausgeglichen und beispielsweise einem möglichen Drift-Effekt vorgebeugt werden sollte (Holmqvist et al., 2011). Im Gegensatz zur früher verwendeten SMI-Toolsuite bietet Tobii ProLab keine Antwortmöglichkeit im Programm an, was einen Workaround notwendig machte: Im Hintergrund wur-

de Microsoft Word ausgeführt, welches ein Dokument bereithielt, in welchem der jeweilige Proband seine Antwort verfassen konnte. Während der Erstellung der Antwort wurden die Augenbewegungen ebenfalls aufgezeichnet und dazu noch ein Screenrecord erstellt. Bei den Antworten wurde darüber hinaus auch ein Word-Count durchgeführt, der zeigen sollte, inwiefern sich die Probanden in ihrem Antwortverhalten unterscheiden.

4. *Durchführung eines stimulusbasierten Interviews:* Im Anschluss an die Erhebung der Eye-Tracking-Daten wurden den Versuchspersonen die dabei entstandenen Videos vorgeführt. Im Rahmen eines interviewartigen Settings wurden die Teilnehmerinnen und Teilnehmer dazu animiert ihre eigenen Augenbewegungen zu kommentieren und eventuell angewandte Strategien zu erläutern. Gleichzeitig wurde in dieser Phase des Experiments der vorab formulierte Interviewleitfaden abgearbeitet. Die gesamten Interviews wurden als Audiomitschnitt für jede Versuchsperson aufgezeichnet. Besondere Auffälligkeiten in den Aussagen wurden direkt notiert und weiter thematisiert. Aufgrund der teilweise relativ langen Bearbeitungsdauer wurde es den Teilnehmerinnen und Teilnehmern an der Studie freigestellt, ob sie lediglich einzelne Aufzeichnungen oder alle gezeigten Fälle sehen und kommentieren wollten. Die Interviews fanden ohne Zeitverzögerung direkt nach Aufzeichnung der Augenbewegungen statt. Dadurch sollte gewährleistet werden, dass die Eindrücke der Versuchspersonen noch frisch und möglichst unverfälscht waren.

Die Gesamtdauer der einzelnen Datenerhebungen war stark abhängig von individuellen Einflussfaktoren (z.B. die allgemeine und professionelle Programmiererfahrung) der jeweiligen Versuchspersonen und variierte daher verhältnismäßig stark. Sie deckten ein Spektrum von ca. 25 Minuten bis hin zu etwa 45 Minuten ab.

6.3 Datenauswertung

Die Auswertung der im Experiment gesammelten Daten unterteilt sich insgesamt in vier Phasen. In einem ersten Schritt werden die Daten aufbereitet und in ein für die Analyse geeignetes Format gebracht (siehe Absatz 6.3.1). Anschließend werden die demographischen Daten der Versuchspersonen, sowie deren ergänzende Informationen (beispielsweise die Performance bei den Code Reviews) analysiert (siehe Absatz 6.3.2). Eine Analyse der grundlegenden Eye-Tracking-Daten, in deren Rahmen diese deskriptiv beschrieben und Auffälligkeiten zwischen den einzelnen Erfahrungsstufen herausgearbeitet werden, erfolgt im nächsten Schritt (siehe Absatz 6.3.3). Den Abschluss der Datenauswertung bildet die Betrachtung der phasenbasierten Eye-Tracking-Daten, welche die zuvor genannten Variablen im zeitlichen Verlauf des Experiments untersucht (siehe Absatz 6.3.4).

6.3.1 Datenaufbereitung

Die Aufbereitung der Daten orientierte sich an der in Kapitel 5 bei Absatz 5.3.1 beschriebenen Vorgehensweise, berücksichtigte dabei jedoch die zuvor bei Absatz 6.1.1 genannten Veränderungen im Experimentaldesign. Konkrete verlief die Datenaufbereitung dabei folgendermaßen:

Fragebogen: Durch die Verwendung von Tobii Pro Lab mussten die vom Fragebogen stammenden Daten direkt mit dieser Software erfasst werden. Darüber hinaus ergaben sich keine Abweichungen zur Vorgängerstudie.

Eye-Tracking-Daten: Die Aufbereitung der Eye-Tracking-Daten orientierte sich erneut an der von Sharafi et al. (2020, S.3135-3139) beschriebenen vierstufigen Vorgehensweise und griff ebenfalls auf eine Kombination von verschiedenen Softwareprodukten zurück. Durch den Einsatz von Tobii Pro Lab ergaben sich Abweichungen zur zuvor beschriebenen C-Studie. Der Ablauf der Datenfilterung wird nachfolgend beschrieben:

1. *Ungefilterte Rohdaten (First Order Data):* Ein Zugriff auf die ungefilterten Rohdaten des Tobii Spectrums erwies sich im Rahmen der Studie als nicht möglich. Die Experimentalsoftware Tobii ProLab unterzieht die anfallenden Rohdaten bereits einer Vorfilterung und Klassifikation, bevor der Nutzer diese näher betrachten kann.

2. *Gefilterte Rohdaten (Second Order Data):* Seitens Tobii Pro Lab werden verschiedene Filteroptionen zur Ermittlung von *fixations* und *saccades* angeboten. Da im Falle der vorliegenden Studie mit 600Hz aufgezeichnet wurde, wurde auf den standardmäßig verwendeten geschwindigkeitsbasierten Filter (IV/T) zurückgegriffen (Tobii Pro, 2020, 2021). Die dabei erlangten Daten wurden im Verlauf der weiteren Auswertung zum Teil für die Berechnung von *komplexen Eye-Tracking-Metriken (Fourth Order Data)* verwendet.

3. *Grundlegende Eye-Tracking-Metriken (Third Order Data):* Tobii ProLab bietet standardmäßig bereits eine große Anzahl von grundlegenden Eye-Tracking-Metriken an, welche direkt nach Abschluss des Experiments exportiert werden können (Tobii Pro, 2021). Vor dem Export dieser Dateien wurden die entsprechenden Metriken erneut hinsichtlich ihrer Relevanz für die vorab formulierten Hypothesen begutachtet und mit Standardwerken des Eye-Trackings (Duchowski, 2017; Holmqvist & Andersson, 2017, 2017) abgeglichen. Dieser Abgleich bedingte sich durch die geräte- bzw. herstellerspezifische Definition der jeweiligen Metriken und hatte das Ziel, diese mit der vorherigen Studie (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023) vergleichbar zu machen, welche einen SMI-Eye-Tracker verwendet hat.

Tabelle 6.1: Überblick über die im Experiment erzielten Punkte

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett (N=34)	5.000	3.618	4	1	12	11
Experten (n=16)	7.000	3.795	6	1	12	11
Novizen (n=18)	3.222	2.365	2	1	9	8

4. *Komplexe Eye-Tracking-Metriken (Fourth Order Data):* Die Berechnung der komplexen Eye-Tracking-Metriken erfolgte auf Basis der *Second* und *Third Order Data* und wurde in R durchgeführt. Im Falle der vorliegenden Studie spielten in Bezug auf diese Kategorie vorrangig Verhältnisse zwischen einzelnen Variablen (z.B. *fixation rate, proportion of dwell time on erroneous lines, ...*), sowie phasenbasierte Metriken (z.B. *number of fixations during phase 1, ...*) eine entscheidende Rolle. Diese wurden für jedes zu reviewende Codebeispiel einzeln und kumuliert über das gesamte Experiment hinweg berechnet. Die entsprechenden R-Skripte zur Filterung und Berechnung finden sich im digitalen Anhang dieser Dissertation.

Interview: Die Interviewdaten der Versuchspersonen wurden erneut zur Verifizierung der verwendeten Strategien genutzt. Sie dienten auch in dieser Studie zur Absicherung der Eye-Tracking-Daten. Sie wurden nicht transkribiert, jedoch im Bedarfsfall als Audioaufzeichnung in die weitere Analyse einbezogen.

6.3.2 Analyse der demographischen Daten

Die aus dem Fragebogen gewonnenen Daten wurden im Verlauf der Analyse vorrangig dazu genutzt um erfahrungsspezifische Unterschiede zwischen den Gruppen der fortgeschrittenen Programmierer bzw. Experten und Novizen zu analysieren. So wird in den nachfolgenden Abschnitten beschrieben, wie sich die *allgemeine* und *professionelle Programmiererfahrung* auf die *Anzahl der im Experiment erzielten Punkte* auswirken (siehe Absatz 6.3.2). Weiterhin werden die Variablen zur *Programmiererfahrung* mit der *Selbsteinschätzung* der Versuchspersonen in Verbindung gebracht (siehe 6.3.2).

Überblick über die im Experiment erzielten Punkte

Zur Messung der Leistung wurden die von den Versuchspersonen abgegebenen Antworten korrigiert und bepunktet. Übergreifend über alle Codebeispiel konnten die Teilnehmerinnen und Teilnehmer im Experiment zwölf Punkte erreichen. Dieser Punktwert wurde allerdings nur von der Gruppe der Fortgeschrittenen und Experten erreicht. Eine aufgabenübergreifende Übersicht über das Abschneiden der Probanden kann der nachfolgenden Tabelle 6.1 entnommen werden:
Es zeigt sich deutlich, dass die Gruppe der fortgeschrittenen Programmierer und

Experten besser abgeschnitten hat. Mit durchschnittlich sieben von zwölf möglichen Punkten konnte diese Gruppe mehr als die doppelte Menge an Punkten erzielen als die Novizinnen und Novizen. Zur Überprüfung dieser Auffälligkeit wurde ein t-Test durchgeführt. Dieser zeigt, dass der Unterschied bezüglich der erzielten Punkte zwischen der Gruppe der fortgeschrittenen Programmierer und Experten (M=7.000; SD=3.795) im Vergleich zur Gruppe der Novizen (M=3.222; SD=2.365) signifikant ist (t(24.565)=-3.433; p=0.000; d=1.212). Nach Cohen (1992) ist dieser Unterschied als groß zu bewerten.
Weiterhin lassen sich zwischen der *Anzahl der erzielten Punkten* und der *allgemeinen* (r_{BP}=.623; p=.000), sowie der *professionellen Programmiererfahrung* (r_{BP}=.597; p=.000) Zusammenhänge beobachten. Diese legen nahe, dass die individuelle Erfahrung der Versuchspersonen ein entscheidender Einflussfaktor auf die erfolgreiche Durchführung eines Code Reviews ist.
Eine gruppenübergreifende Betrachtung der deskriptiven Statistik (siehe Tabelle 6.2) zu den einzelnen Aufgaben zeigt weiterhin, dass vor allem bei den Aufgaben zum *Size of dynamic array* und *Missing copy constructor* relativ schlechte Ergebnisse erzielt wurden. Dies mag darauf zurückzuführen sein, dass diese Aufgaben mehrere Fehler enthielten, welche von den Versuchspersonen unter Umständen übersehen wurden bzw. die Bearbeitung bereits nach der Entdeckung des ersten Fehlers eingestellt wurde.

Tabelle 6.2: Gruppenübergreifende Betrachtung der Aufgaben

Aufgabe	Mittelwert	SD	Mdn
Stack error	0.441	0.540	0.000
Non-virtual destructor	0.471	0.507	0.000
Missing method override	0.441	0,504	0.000
Slicing problem	0.588	0.500	1.000
Korrektes Codebeispiel 1	0.441	0.504	0.000
Size of dynamic array	1.000	1.073	1.000
Missing copy constructor	1.088	1.055	1.000
Korrektes Codebeispiel 2	0.529	0.507	1.000

Des Weiteren kann aus Tabelle 6.2 entnommen werden, dass bei den Aufgaben *Stack error* (M=0.441; SD=0.540) und *Korrektes Codebeispiel 1* (M=0.441; SD=0.500) die wenigsten Punkte erzielt wurden. Gegenteilig verhält sich hier die vierte zu bearbeitende Aufgabe. Hier schnitten die Probanden mit einem Mittelwert von 0.588 (SD=0.500) am besten ab.

Selbsteinschätzung der Versuchspersonen

Ein weiterer Teil der demographischen Daten, welcher in Hinblick auf die Annahmen der Expertiseforschung relevant war, war die Selbsteinschätzung der Versuchspersonen hinsichtlicher ihrer eigenen Fähigkeiten zur Durchführung eines Code Reviews.

Abgefragt wurden dabei die Einschätzung der allgemeinen Programmierkenntnisse, die Programmierkenntnisse in C++ und die Fähigkeit, ein Review durchzuführen. Diese Einschätzung erfolgte über drei Fragen, welche mit einer vierstufigen Likert-Skala beantwortet werden konnten. Die Fragen lauteten folgendermaßen:

- *Self1:* Schätzen Sie Ihre allgemeinen Programmierkenntnisse als gut ein?
- *Self2:* Schätzen Sie Ihre Programmierkenntnisse in C++ als gut ein?
- *Self3:* Schätzen Sie Ihre Fähigkeiten als Reviewer als gut ein?

Die deskriptive Statistik zur Beantwortung dieser Fragen wird in der nachfolgenden Tabelle 6.3 dargestellt:

Tabelle 6.3: Selbsteinschätzung der Probanden

Gruppe	Item	Mittelwert	SD	Mdn
Komplett	Self1	2.971	0.521	3.000
	Self2	2.412	0.783	2.000
	Self3	2.294	0.579	2.000
Experten	Self1	3.250	0.447	3.000
	Self2	2.688	0.873	3.000
	Self3	2.375	0.619	2.000
Novizen	Self1	2.722	0.461	3.000
	Self2	2.167	0.618	2.000
	Self3	2.222	0.548	2.000

Im Vergleich der beiden Erfahrungsstufen zeigt sich, dass die erfahrenere Gruppe die eigenen Fähigkeiten über alle Fragen hinweg als besser einschätzt. Im Falle von *Self1* zeigt ein t-Test statistisch signifikante Unterschiede an. So schätzen Experten und fortgeschrittene Programmierer (M=3.250; SD=0.447) im Vergleich zu Novizinnen und Novizen (M=2.722; SD=0.461) ihre allgemeinen Programmierkenntnisse als signifikant besser ein (t(31.735)=-3.386; p=0,002; d=1.161). Auf Basis von Cohens Einschätzungen (1992) kann dieser Unterschied als sehr ausgeprägt betrachtet werden.
Weiterhin beurteilen erfahrene Versuchspersonen (M=2.688; SD=0.873) ihre Programmierkenntnisse in C++ als besser, als Novizinnen und Novizen (M=2.167; SD=0.618) (t(26.676)=-1.984; p=0.058; d=0.700). Ebenso geht die Gruppe der Fortgeschrittenen/Experten davon aus, dass sie über bessere Fähigkeiten als Reviewer verfügen (M=2.375; SD=0.0619), als Novizinnen und Novizen (M=2.222; SD=0.548), zeigen hier aber eine verhältnismäßige Einschätzung (t(30.238)=-0.758; p=0.455; d=1.161). Es bleibt jedoch anzumerken, dass sowohl *Self2*, als auch *Self3* die statistische Signifikanz verfehlen.
Ergänzend zu den Unterschieden zwischen den beiden Erfahrungsstufen wurden auch die Korrelationen zwischen der Selbsteinschätzung und der angesammelten allgemeinen und professionellen Programmiererfahrung ermittelt. Diese Korrelationen werden in der nachfolgenden Tabelle 6.4 dargestellt:

Tabelle 6.4: Korrelationskoeffizienten zur Programmiererfahrung und Selbsteinschätzung

Korrelationskoeffizienten	Self1	Self2	Self3
Allgemeine Programmiererfahrung in Jahren	r_{BP}=0.400 (p=0.019)	r_{BP}=0.390 (p=0.023)	r_{BP}=0.299 (p=0.086)
Professionelle Programmiererfahrung in Jahren	r_{BP}=0.427 (p=0.012)	r_{BP}=0.277 (p=0.112)	r_{BP}=0.256 (p=0.144)

6.3.3 Deskriptive Betrachtung der Eye-Tracking-Daten

Die nachfolgenden Absätze basieren auf einer kumulierten Betrachtung der relevanten Eye-Tracking-Metriken der von Sheridan und Reingold (2017, S.5) dargestellten *holistic models of image perception*. Die Vorgehensweise unterteilte sich dabei in eine deskriptive Beschreibung der jeweiligen Variable, sowie in die Analyse von erfahrungsbedintgen Unterschieden zwischen Experten/Fortgeschrittenen und Novizinnen und Novizen. Analysiert wurden dabei die *total* und *average number of fixations*, die *total* und *average duration of fixation*, die *fixation rate*, die *number of saccades*, die *number of visits on error*, sowie die *total dwell time on error*.

Deskriptive Statistik zur *total number of fixations*

Tabelle 6.5: Deskriptive Statistik zu *total number of fixations*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	717.735	512.820	526.000	145.000	2202.000	2057.000
Experten	778.125	500.421	635.000	145.000	2202.000	2057.000
Novizen	664.056	532.029	448.000	216.000	2081.000	1865.000

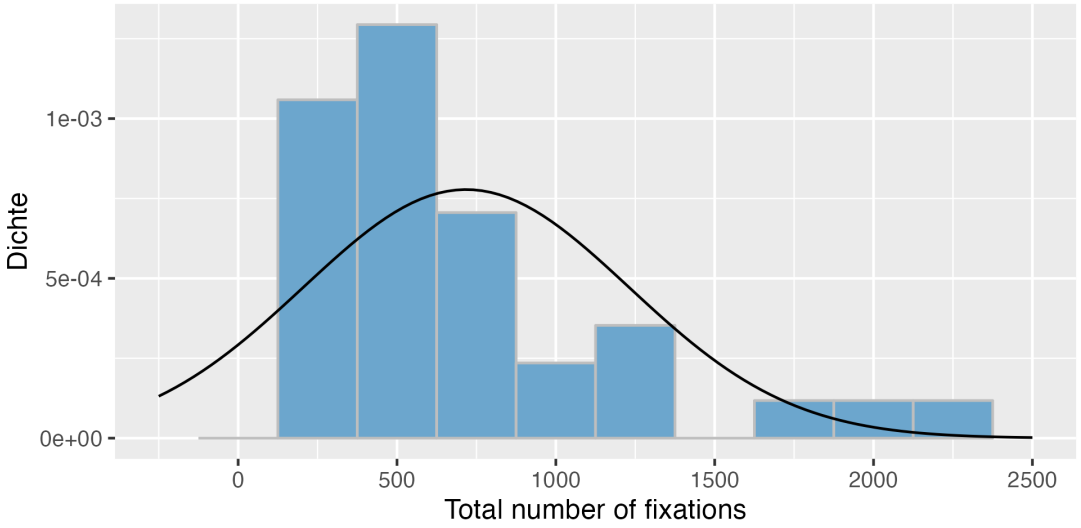

Abbildung 6.4: Verteilung der *total number of fixations*

Tabelle 6.5 bildet die relevante deskriptive Statistik zur über alle Aufgaben des Ex-

periments übergreifenden *total number of fixations* ab. Es zeigt sich, dass die Experten einen höheren Mittelwert (M=778.125; SD=500.421) und eine größere Spannweite (2057.000) im Vergleich zu den Novizen (M=664.056; SD=532.029; Spannweite=1865.000) aufweisen. Aufgrund von statistischen "Ausreißern" in beiden Gruppen stellt sich jedoch der Median als die verlässlichere Angabe dar. Für die Fortgeschrittenen/Experten beläuft sich dieser auf 635.000 und für die Novizen auf 448.000.
Die *total number of fixation* wurde hinsichtlich ihrer Verteilung untersucht. Ein Kolmogorov-Smirnov- (D=1; p=0.000) und ein Saphiro-Wilk-Test (W=.829; p=0.000) bestätigen, dass keine Normalverteilung vorliegt. Ergänzend dazu und auf Grund der Kritik an diesen Tests (Cohen, 1988; Cohen, 1992; Field & Field, 2012) wurde zudem anhand eines Histograms (siehe Abbildung 6.4 grafisch überprüft, ob eine Normalverteilung vorliegt. Das Resultat bestätigte die beiden durchgeführten Tests.
Es wurde weiterhin untersucht, ob sich Zusammenhänge zwischen der *total number of fixation* und verschiedenen begleitenden Daten ergaben. Konkret wurden Korrelationen zwischen der *allgemeinen* und *beruflichen Programmiererfahrung*, sowie der insgesamt erzielten Punktzahl im Experiment untersucht. Zwischen der *total number of fixations* und der *allgemeinen Programmiererfahrung* kann eine schwache Korrelation von r_{BP}=-.133 beobachtet werden, die jedoch keine statistische Signifikanz erreicht (p=.455). Ähnlich verhält es sich im Fall der *beruflichen Programmiererfahrung*. Hier trat ebenfalls eine schwache (r_{BP}=-.141), statistisch nicht signifikante (p=.428) Korrelation auf. Deutlich schwächer fällt der Zusammenhang zwischen der *Anzahl der erzielten Punkte* und der *total number of fixations* aus. Es ergab sich ein statistisch nicht signifikanter (p=.785) Wert von r_{BP}=.049 (Cohen, 1988; Cohen, 1992).
Weiterhin wurden Unterschiede zwischen den beiden Gruppen der Fortgeschrittenen und Novizen berechnet. Ein Mann-Whitney-U-Test konnte keine signifikanten Unterschiede nachweisen (U=108; z=1.217; r=.209; p=.224).

Deskriptive Statistik zur *average number of fixations*

Tabelle 6.6: Deskriptive Statistik zu *average number of fixations*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	89.717	64.102	65.750	18.125	275.250	257.125
Experten	97.266	62.553	79.375	18.125	275.250	257.125
Novizen	83.007	66.504	56.000	27.000	260.125	233.125

Hinsichtlich der *average number of fixations* konnte ein ähnliches Bild wie bei der *total number of fixations* beobachtet werden. Die Werte fallen im Durchschnitt für die Fortgeschrittenen/Experten (M=97.266; SD=62.553; Mdn=79.375) höher aus, als für die Novizen (M=83.007; SD=66.504; Mdn=56.000).
Bezüglich der Verteilung ergaben sich keine Unterschiede zur zuvor beschriebenen *total number of fixations*. Der Kolmogorov-Smirnov-Test (D=1; p=0.000) und Saphiro-Wilk-Test (W=0.829; p=0.000) bestätigen, dass die *average number of fixations* nicht nor-

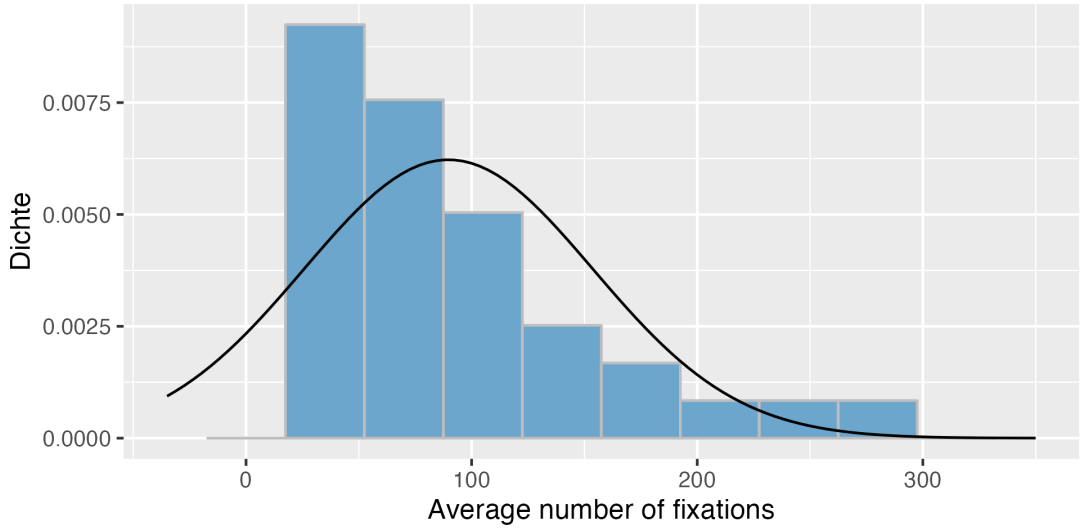

Abbildung 6.5: Verteilung der *average number of fixations*

malverteilt ist. Die grafische Überprüfung (siehe Abbildung 6.5) bestätigt weiterhin die Testresultate.
Das gleiche Bild wie bei der *total number of fixations* zeigt sich bei den berechneten Korrelationen. Es kann eine schwache, nicht signifikante Korrelation (r_{BP}=-0.133; p=0.455 zwischen der *average number of fixations* und der *allgemeinenen Programmiererfahrung* beobachtet werden. Auch die Korrelation hinsichtlich der *beruflichen Programmiererfahrung* zeigt ein identisches Resultat (r_{BP}=-0.141; p=0.428), welches statistisch nicht signifikant wird. Für die *Anzahl der erzielten Punkte* ergibt sich ebenfalls die nicht signifikante (p=0.785) Korrelation von r_{BP}=0.049 (Cohen, 1988; Cohen, 1992).
Ergänzend dazu wurden die beiden Gruppen hinsichtlich signifikanter Unterschiede untersucht. Ein Mann-Whitney-U-Test konnte keine signifikanten Unterschiede nachweisen (U=108; z=1.217; r=0.209; p=0.224).

Deskriptive Statistik zur *total fixation duration*

Tabelle 6.7: Deskriptive Statistik zu *total fixation duration in [ms]*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	383180.100	234681.900	298495.500	126588.000	1122145.000	995557.000
Experten	402345.600	237106.300	365297.000	142989.000	1122145.000	979156.000
Novizen	366144.100	238019.200	257315.000	126588.000	1044255.000	917667.000

Ähnlich zur zuvor beschriebenen *total* und *average number of fixations* verhält es sich mit der *total fixation duration* welche die summierte, aufgabenübergreifende Zeit in Millisekunden darstellt, die seitens der Probanden für Fixations aufgewandt worden ist. Es lässt sich beobachten, dass diese für die Fortgeschrittenen/Experten (M=402345.000; SD=237106.300; Mdn=365297.000) länger ausfällt, als für die Gruppe der Novizinnen und Novizen (M=366144.100; SD=238019.200).

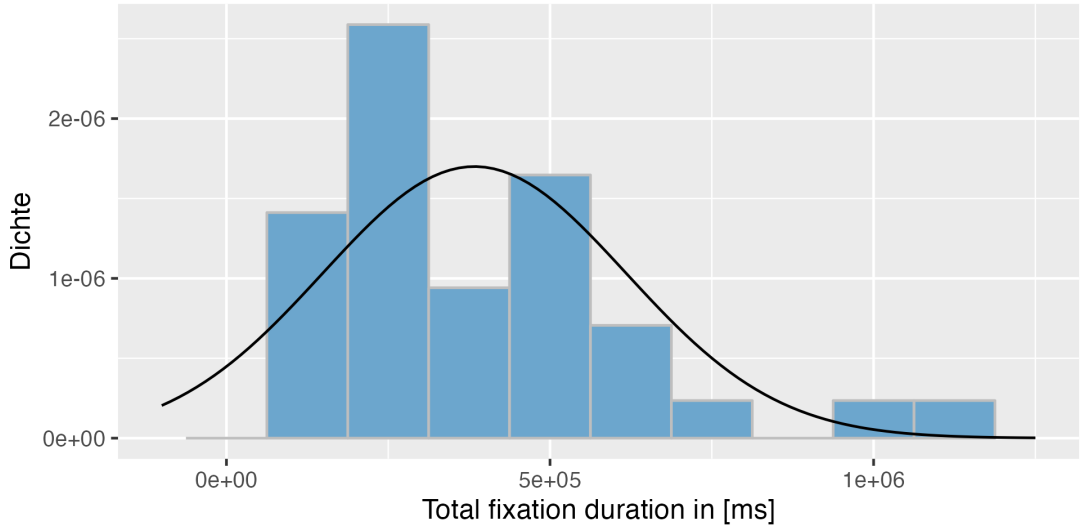

Abbildung 6.6: Verteilung der *total fixation duration*

Auch im Falle der *total fixation duration* liegt keine Normalverteilung vor. Dies wurde erneut durch einen Kolmogorov-Smirnov-Test (D=1, p=.000) und einen Saphiro-Wilk-Test (W=.829, p=.000) abgesichert. Eine weitere Absicherung erfolgt durch die Begutachtung eines entsprechenden Histograms (siehe 6.6).
Hinsichtlich möglicher Korrelationen zwischen der *total fixation duration* und der *allgemeinen* (r_{BP}=-.136; p=.442), sowie der beruflichen Programmiererfahrung (r_{BP}=-.137; p=.438) ergeben sich lediglich schwache Zusammenhänge, die keine statistische Signifikanz erreichen. Selbiges lässt sich für die Korrelation zwischen der *Anzahl der erzielten Punkte* und der *total fixation duration* beobachten (r_{BP}=-.085; p=.633).
In einem weiteren Schritt wurden mittels eines Mann-Whitney-U-Test mögliche Unterschiede zwischen den beiden Gruppen untersucht. Es konnte ein schwacher Effekt beobachtet werden, der jedoch keine statistische Signifikanz erreicht (U=118; z=.871; r=.149; p=.384).

Deskriptive Statistik zur *average fixation duration*

Tabelle 6.8: Deskriptive Statistik zu *average fixation duration in [ms]*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	579.957	132.145	539.427	358.963	986.131	627.168
Experten	557.632	147.415	518.573	369.073	986.131	617.058
Novizen	599.802	117.631	627.192	358.963	764.169	405.206

Bei der *average fixation duration in [ms]* zu, lässt sich eine Trendumkehr beobachten: Die Gruppe der Experten zeigt hier kleinere Werte (M=557.632; SD=147.415; Mdn=518.573) als die Novizinnen und Novizen (M=599.802; SD=117.631; Mdn=627.192).
Auch für die *average fixation duration* lässt sich keine Normalverteilung beobach-

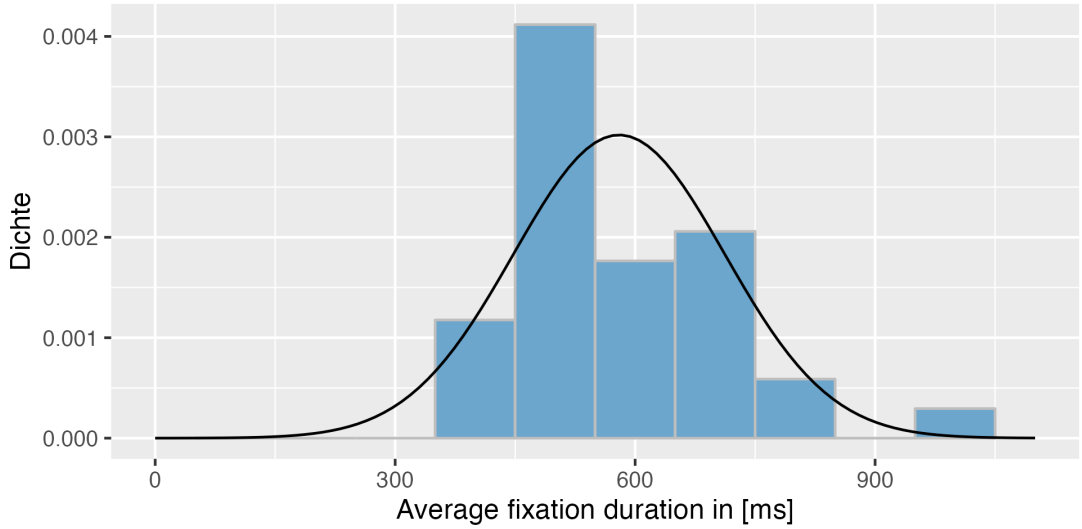

Abbildung 6.7: Verteilung der *average fixation duration*

ten. Der Kolmogorov-Smirnov- (D=1; p=0.000) und der Saphiro-Wilk-Test (W=0.944; p=0.083) liefern keine Anzeichen dafür, dass die Variable normalverteilt ist. Die grafische Darstellung durch ein Histogram (siehe Abbildung 6.7) unterstreicht diese Resultate.

In Bezug auf mögliche Korrelationen zwischen der *average fixation duration* und der *allgemeinen* (r_{BP}=-.039; p=.827), sowie der *professionellen Programmiererfahrung* (r_{BP}=-.051; p=.777) ergeben sich lediglich schwache, statistisch nicht signifikante Zusammenhänge. Etwas stärker ausgeprägt, aber dennoch nicht statistisch signifikant zeigt sich der Korrelationskoeffizient von *average fixation duration* und der *Anzahl der erzielten Punkte* (r_{BP}=-.273; p=.118).

Abschließend wird mittels Mann-Whitney-U-Test untersucht, ob sich zwischen den Experten und Novizen signifikante Unterschiede ergeben. Der Test liefert keine statistisch signifikanten Ergebnisse (U=182; z=1.286; r=.221; p=.199). Somit kann davon ausgegangen werden, dass sich fortgeschrittene Programmierer (Mdn=518.573) bezüglich ihrer *average fixation duration* nicht signifikant von Anfänger*innen unterscheiden (Mdn=627.192).

Deskriptive Statistik zur *fixation rate*

Tabelle 6.9: Deskriptive Statistik zur *fixation rate*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1.504	0.365	1.507	0.838	2.534	1.696
Experten	1.614	0.317	1.644	0.838	2.135	1.297
Novizen	1.407	0.385	1.226	1.035	2.534	1.498

Die Betrachtung der deskriptiven Statistik zur *fixation rate* deutet erneut leichte Unterschiede zwischen den Experten und Novizen an. So beläuft sich der Mittelwert für

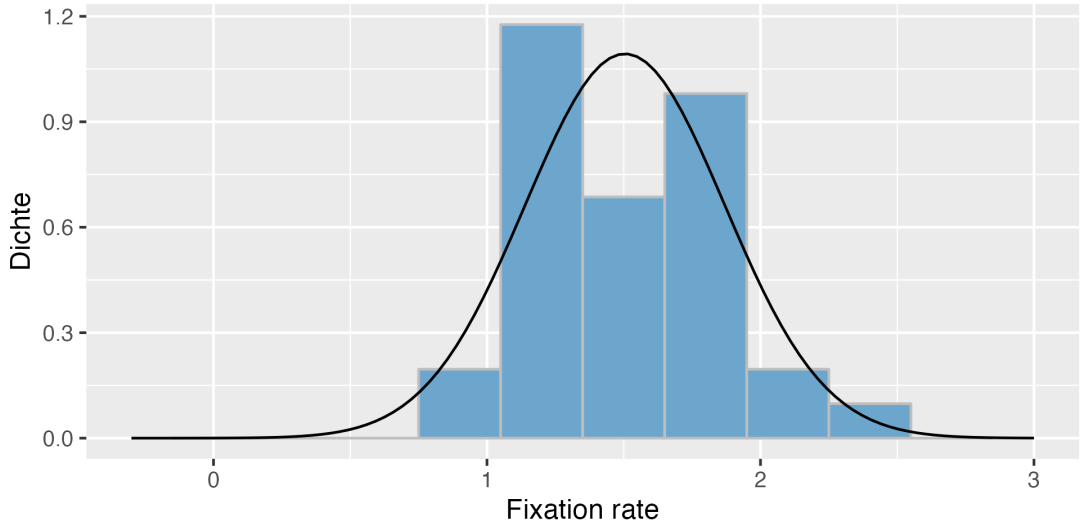

Abbildung 6.8: Verteilung der *fixation rate*

die Gruppe der Experten auf 1.614 fixations/sec (SD=.317), wohingegen die weniger erfahrenen Versuchspersonen einen Mittelwert von 1.407 fixations/sec (SD=.385) erzielen. Wie auch bei den bereits zuvor beschriebenen Eye-Tracking-Metriken erweist sich der Median aufgrund der Ausreißer als belastbarer. Dieser beläuft sich für die Experten auf 1.644 fixations/sec und für die Novizen auf 1.226 fixations/sec.
Auch die *fixation rate* wurde hinsichtlich ihrer Verteilung untersucht. Die Ergebnisse des Kolmogorov-Smirnov-Test (D=.820; p=.000) und des Shapiro-Wilk-Test (W=.959; p=.225) liefern keine eindeutige Auskunft darüber, ob von einer Normalverteilung ausgegangen werden kann. Das zur Absicherung erstellte Histogram (siehe 6.8) legt nahe, dass in diesem Fall keine Normalverteilung der Daten vorliegt.
Weiterhin wurde untersucht, ob sich zwischen der *fixation rate* und der angesammelten Erfahrung, sowie der im Experiment erzielten Anzahl an Punkten Zusammenhänge beobachten lassen. Zwischen der *fixation rate* und der *allgemeinen* (r_{BP}=.038; p=.832), sowie der *professionellen Programmiererfahrung* (r_{BP}=.073; p=.683) lassen sich keine relevanten Korrelationen beobachten. Ein anderes Bild präsentiert sich hingegen bezüglich des Zusammenhangs zwischen der *fixation rate* und der *Anzahl der erzielten Punkte*. An dieser stelle ergibt sich eine signifikante Korrelation mit einer mittleren Effektstärke (r_{BP}=.361; p=.036) (Cohen, 1988; Cohen, 1992).
In einem weiteren Schritt wurde untersucht ob sich die beiden Gruppen hinsichtlich der *fixation rate* signifikant voneinander unterscheiden. Ein Mann-Whitney-U-Test kann signifikante Unterschiede zwischen den Experten und Novizen belegen (U=86; z=1.993; r=.342; p=.046) (Cohen, 1988; Cohen, 1992). Es lässt sich somit festhalten, dass Experten (Mdn=1.644) eine signifikant höhere *fixation rate* bei Code Reviews aufweisen, als Novizen (Mdn = 1.226) (Cohen, 1988; Cohen, 1992)..

Tabelle 6.10: Deskriptive Statistik zur *total number of saccades*

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1892.000	1025.882	1979.5	276	4514	4238
Experten	1564.562	1136.941	1243.0	276	4514	4238
Novizen	2183.056	843.678	2178.5	535	3720	3185

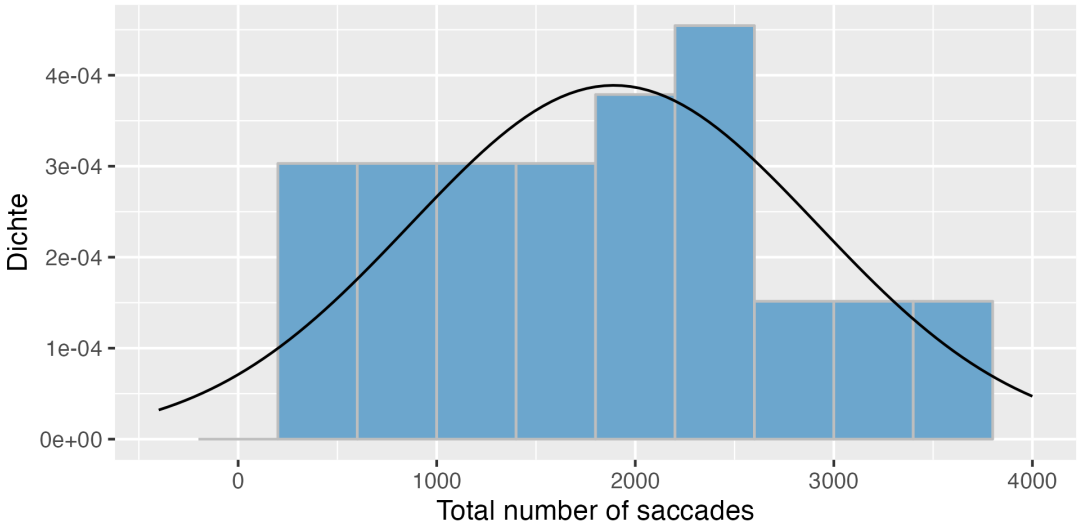

Abbildung 6.9: Verteilung der *total number of saccades*

Deskriptive Statistik zur *total number of saccades*

Auch die auf den Saccaden basierenden Metriken wurden im Kontext der Datenauswertung analysiert. In der deskriptiven Statistik zur *total number of saccades* lassen sich ebenfalls Unterschiede zwischen Experten und Novizen erkennen. So zeigt sich, dass die erfahreneren Versuchspersonen (M=1564.562; SD=1136.941; Mdn=1243.000) über das gesamte Experiment hinweg weniger Saccaden zeigen, als beispielsweise die Gruppe der Novizen (M=2183.056; SD=843.678; Mdn=2178.500).
Das Vorliegen einer Normalverteilung wurde erneut mittels eines Kolmogorov-Smirnov-Test (D=1.000; p=.000) und eines Shapiro-Wilk-Test (W=.972; p=.511) untersucht. Erneut liefern die Tests an dieser Stelle keine eindeutigen Ergebnisse. Das zur Absicherung erstellte Histogramm (siehe 6.9) lässt darauf schließen, dass keine Normalverteilung für die Variable *total number of saccades* vorliegt.
Zusammenhänge zwischen der *total number of saccades* und der *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der erzielten Punkte* wurden ebenfalls betrachtet. Sowohl für die *allgemeine* (r_{BP}=-.292; p=.093) und die *professionelle Programmiererfahrung* (r_{BP}=-.252; p=.150) zeigen sich schwache Korrelationen zur *total number of saccades*, die jedoch keine statistische Signifikanz erreichen. Anders verhält sich die Korrelation zwischen der *total number of saccades* und der *Anzahl der erzielten Punkte*. Diese erreicht ein starkes Niveau und statistisches Signifikanz (r_{BP}=-.610; p=.000) (Cohen, 1988; Cohen, 1992).
Abschließend wurde über einen Mann-Whitney-U-Test überprüft, ob sich zwischen

den beiden Gruppen hinsichtlich der *total number of saccades* statistisch signifikante Unterschiede ergeben. Der Test kann diese belegen (U=205; z=2.101; r=.360; p=.036). Somit kann festgehalten werden, dass Experten (Mdn=1243.000) die Code Reviews mit einer signifikant kleineren *total number of saccades* absolvieren, als die Novizen (Mdn=2178.000).

Deskriptive Statistik zur *average number of saccades*

Tabelle 6.11: Deskriptive Statistik zur average number of saccades

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	236.500	128.235	247.438	34.500	564.250	529.750
Experten	195.570	142.118	155.375	34.500	564.250	529.750
Novizen	272.882	105.460	272.312	66.875	465.000	398.125

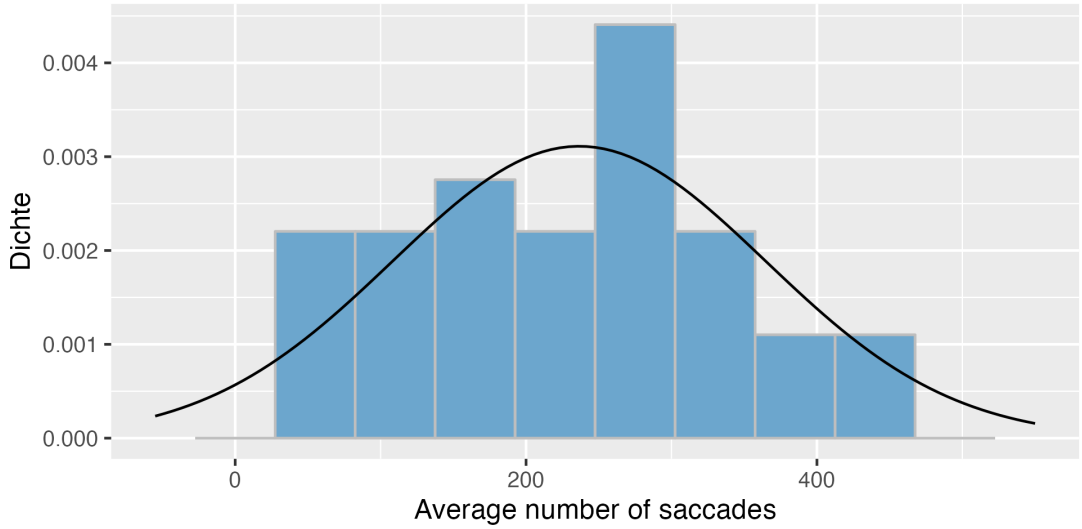

Abbildung 6.10: Verteilung der *average number of saccades*

Bei der *average number of saccades* zeigt sich das gleiche Bild wie bei der zuvor beschriebenen *total number of saccades*. Die Experten weisen einen Mittelwert von 195.570 (SD=142.118) und einen Median von 155.375 auf. Für die Novizen fallen der Mittelwert (272.882; SD=105.460) und der Median (272.312) höher aus.
Es wurde erneut mittels eines Kolmogorov-Smirnov-Test (D=1.000; p=.000) und eines Shapiro-Wilk-Test (W=.972; p=.511) untersucht ob die *average number of saccades* normalverteilt ist. Die Testergebnisse sind an dieser Stelle unklar und bieten keine eindeutigen Lösung. Erneut wird zur Absicherung ein Histogramm erstellt (siehe 6.10), welches darauf schließen lässt, dass keine Normalverteilung für die Variable *average number of saccades* vorliegt.
Mögliche Zusammenhänge zwischen der *average number of saccades* und der *allgemeinen* und *professionellen Programmiererfahrung*, sowie der *Anzahl der erzielten Punkte* wurden ebenfalls betrachtet. Dabei zeigen sich für die *allgemeine* (r_{BP}=-.292; p=.093)

und die *professionelle Programmiererfahrung* (r_{BP}=-.252; p=.150) schwache Korrelationen zur *total number of saccades*, welche aber keine statistische Signifikanz erreichen. Identisch zur *total number of saccades* verhält sich die Korrelation zwischen der *total number of saccades* und der *Anzahl der erzielten Punkte*. Diese erreicht ebenfalls ein starkes Niveau und ist statistisch signifikant (r_{BP}=-.610; p=.000) (Cohen, 1988; Cohen, 1992).
Unterschiede zwischen den beiden Gruppen wurden über einen Mann-Whitney-U-Test analysiert. Der Test kann belegen (U=205; z=2.101; r=.360; p=.036), dass Experten (Mdn=1243.000) die Code Reviews mit einer signifikant kleineren *average number of saccades* durchlaufen, als die Novizen (Mdn=2178.000).

Deskriptive Statistik zur *total number of visits on erroneous lines*

Tabelle 6.12: Deskriptive Statistik zur total number of visits on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	41.118	46.459	26.000	0	196	196
Experten	39.875	40.905	28.500	0	174	174
Novizen	42.222	52.066	20.000	6	196	190

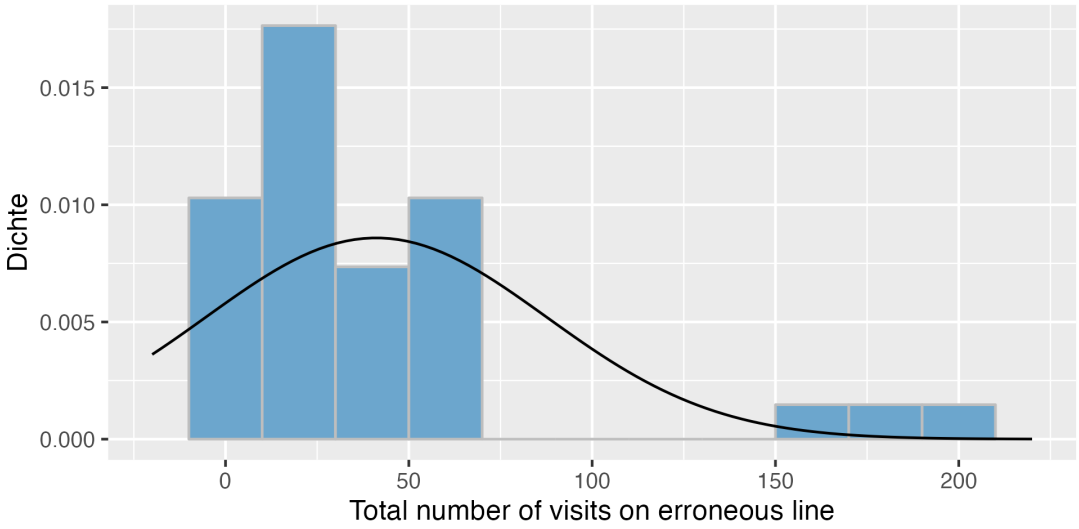

Abbildung 6.11: Verteilung der *total visits on erroneous line*

In Bezug auf die deskriptive Statistik zu den AOI-basierten Eye-Tracking-Metriken wurde zuerst die *total number of visits on erroneous lines* betrachtet. Die Mittelwerte zeigen für diese Variable nur marginale Unterschiede zwischen den Experten (M=39.875; SD=40.905) und Novizen (M=42.222; SD=52.066) an. Ausgeprägtere und bereinigte Ergebnisse lassen sich am Median ablesen. Für die Experten beläuft sich dieser auf 28.500, für die Novizen auf 20.000.
Der Kolmogorov-Smirnov- (D=.971; p=.000) und der Shapiro-Wilk-Test (W=.701; p=.000) deuten darauf hin, dass im falle der *total number of visits on erroneous lines* kei-

ne Normalverteilung vorliegt. Diese Ergebnisse werden auch durch das in Abbildung 6.11 dargestellte Histogramm verdeutlicht. Es kann davon ausgegangen werden, dass für die Variable *total number of visits on erroneous lines* keine Normalverteilung vorliegt. Korrelationen zur *allgemeinen* (r_{BP}=-.125; p=.483) und *professionellen Programmiererfahrung* (r_{BP}=-.149; p=.399), sowie zur *Anzahl der erzielten Punkte* (r_{BP}=.003; p=.987) wurden für die *total number of visits on erroneous lines* ebenfalls analysiert. Dabei zeigten sich jedoch nur schwache Korrelationen, die keine statistische Signifikanz erreichen (Cohen, 1988; Cohen, 1992).
Abschließend wurde durch einen Mann-Whitney-U-Test untersucht, ob sich Unterschiede zwischen den beiden Gruppen ergeben. Der Test liefert jedoch keine relevanten Erkenntnisse (U=129; z=.501; r=.086; p=.617). Somit kann davon ausgegangen werden, dass sich die Experten (Mdn=28.500) und Novizen (Mdn=20.000) bei der Durchführung eines Code Reviews hinsichtlich der *total number of visits on erroneous lines* nicht voneinander unterscheiden.

Deskriptive Statistik zur *average number of visits on erroneous lines*

Tabelle 6.13: Deskriptive Statistik zur average number of visits on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	6.853	7.743	4.333	0	32.667	32.667
Experten	6.646	6.817	4.750	0	29.000	29.000
Novizen	7.037	8.678	3.333	1	32.667	31.667

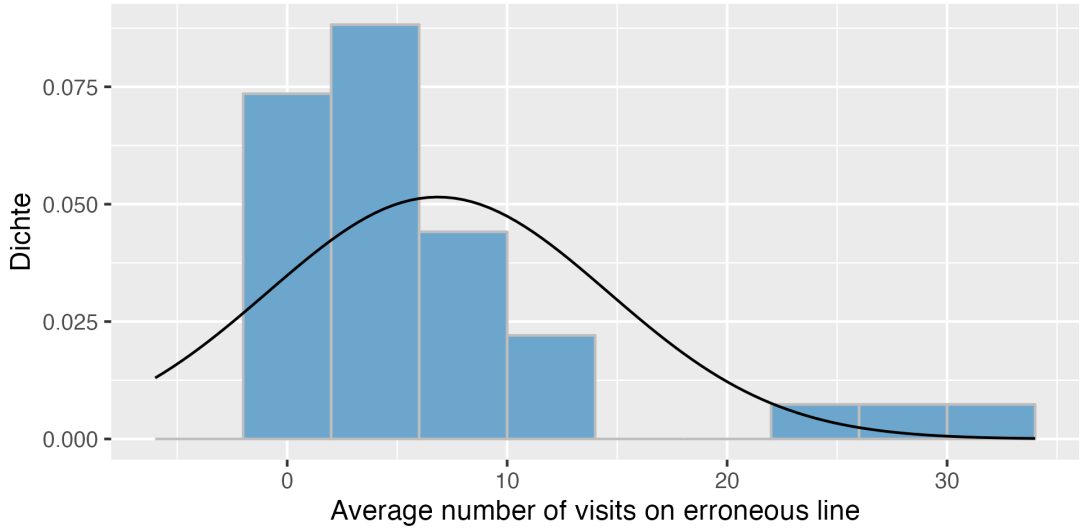

Abbildung 6.12: Verteilung der *average visits on erroneous line*

Die nächste AOI-basierte Metrik, welche untersucht worden ist, stellt die *average number of visits on erroneous lines* dar. Ähnlich zur zuvor beschriebenen *total number of visits on erroneous lines*, zeigen sich auch an dieser Stelle lediglich marginale Unterschiede zwischen den beiden Gruppen. Die Experten weisen einen Mittelwert von 6.646

(SD=6.817) und einen Median von 4.750 auf. Der Mittelwert der Novizen beläuft sich im Gegensatz dazu auf 7.037 (SD=8.678) und der Median auf 3.333.
Die Überprüfung auf Normalverteilung legt ebenfalls nahe, dass diese für die *average number of visits on erroneous lines* nicht gegeben ist. Dies wird durch den Kolmogorov-Smirnov- (D=.845; p=.000) und den Shapiro-Wilk-Test (W=.701; p=.000), sowie durch das in Abbildung 6.12 dargestellte Histogramm bestätigt.
Mögliche Korrelationen zur *allgemeinen* (r_{BP}=-.125; p=.483) und *professionellen Programmiererfahrung* (r_{BP}=-.149; p=.399), sowie zur *Anzahl der erzielten Punkte* (r_{BP}=.003; p=.987) wurden ebenfalls überprüft. Dabei zeigen sich aber wie bei der *total number of visits on erroneous lines* keine nennenswerte Ergebnisse.
Unterschiede zwischen den beiden Gruppen wurden ebenfalls durch einen Mann-Whitney-U-Test (U=129; z=.501; r=.086; p=.617) überprüft. Dieser kann jedoch keine Unterschiede feststellen. Insofern kann auch im Falle der *average number of visits on erroneous lines* davon ausgegangen werden, dass sich Experten (Mdn=4.750) und Novizen (Mdn=3.333) nicht statistisch signifikant voneinander unterscheiden.

Deskriptive Statistik zur *total dwell time on erroneous lines*

Tabelle 6.14: Deskriptive Statistik zur total dwell time on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	23739.820	20513.460	17754.000	4244	86660	82416
Experten	24790.690	19987.770	18147.500	4706	80731	76025
Novizen	22805.720	21502.970	15667.500	4244	86660	82416

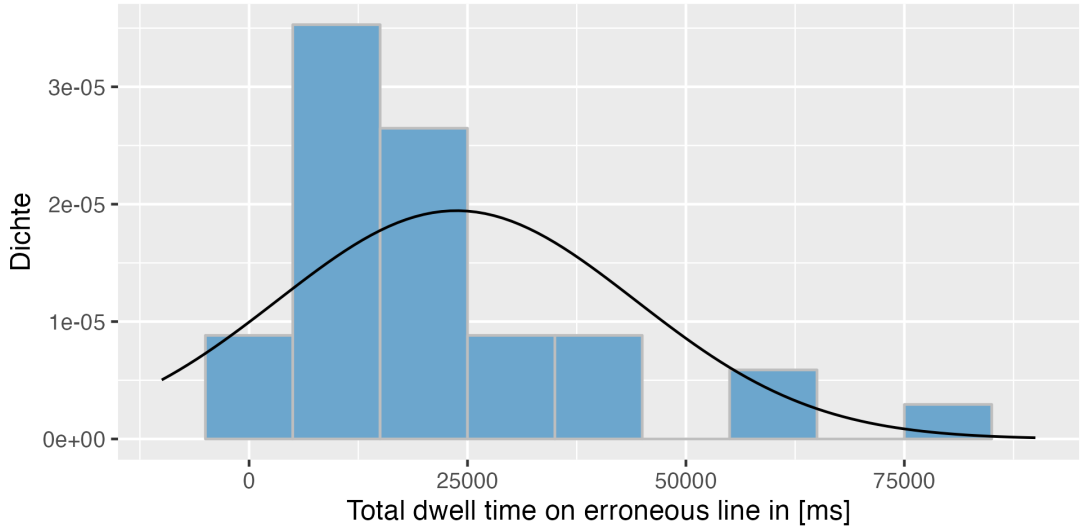

Abbildung 6.13: Verteilung der *total dwell time on erroneous line*

Die *total dwell time on erroneous lines* stellt eine weitere AOI-basierte Eye-Tracking-Metrik dar, die untersucht worden ist. Erneut lassen sich bezüglich der Mittelwerte und Mediane leichte Unterschiede zwischen den Experten und Novizen ablesen. Die

Experten zeigen einen Mittelwert von 24790.690 ms (SD=19987.770 ms) und einen Median von 18147.500 ms, wohingegen diese Werte für die Novizen niedriger ausfallen (M=22805.720 ms; SD=21502.970 ms; Mdn=15667.500 ms). Insgesamt zeigt sich anhand der deskriptiven Statistik, dass die erfahreneren Versuchspersonen eine längere *total dwell time* für die fehlerhaften Zeilen aufweisen, als Anfängerinnen und und Anfänger.
Erneut wird über die zuvor genannten Testverfahren überprüft, ob eine Normalverteilung vorliegt. Sowohl der Kolmogorov-Smirnov- (D=1; p=.000), als auch der Shapiro-Wilk-Test (W=.778; p=.000) bestätigen, dass die *total dwell time on erroneous lines* nicht normalverteilt ist. Selbiges wird auch durch das Histogramm in Abbildung 6.13 bestätigt.
Mögliche Zusammenhänge zwischen der *total dwell time on erroneous lines* und der angesammelten Erfahrung der Versuchspersonen, sowie mit der *Anzahl der erzielten Punkte* wurden ebenfalls betrachtet. Es konnten keine statistisch signifikanten Korrelationen für die *allgemeine Programmiererfahrung* (r_{BP}=-.129; p=.467), die *professionelle Programmiererfahrung* (r_{BP}=-.155; p=.380) oder die *Anzahl der erzielten Punkte* (r_{BP}=.041; p=.818) beobachtet werden.
Abschließend wird auch für die *total dwell time on erroneous lines* untersucht, ob sich zwischen den beiden Gruppen statistisch signifikante Unterschiede ergeben. Ein Mann-Whitney-U-Test (U=123; z=.699; r=.120; p=.484) zeigt auf, dass sich die Experten (Mdn=18147.500) nicht signifikant von den Novizen (Mdn=15667.500) bezüglich ihrer *total dwell time on erroneous lines* unterscheiden.

Deskriptive Statistik zur *average dwell time on erroneous lines*

Tabelle 6.15: Deskriptive Statistik zur average dwell time on erroneous lines

Gruppe	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	3956.637	3418.909	2959.000	707.333	14443.330	13736.000
Experten	4131.781	3331.295	3024.583	784.333	13455.170	12670.830
Novizen	3800.954	3583.829	2611.250	707.333	14443.333	13736.000

Die letzte relevante Eye-Tracking-Metrik die in diesem Kontext betrachtet wird, stellt die *average dwell time on erroneous lines* dar. Diese fällt für die Experten (M=4131.781 ms; SD=3331.295 ms; Mdn=3023.583 ms) im Schnitt länger aus, als für die Novizen (M=3800.954 ms; SD=3583.829 ms; Mdn=2611.250 ms).
Hinsichtlich der Verteilung zeigt sich das selbe Bild wie für die *total dwell time on erroneous lines*: Der Kolmogorov-Smirnov-Test (D=1; p=.000), der Shapiro-Wilk-Test (W=.778; p=.000) und das Histogramm (siehe Abbildung 6.14) belegen, dass die *average dwell time on erroneous lines* nicht normalverteilt ist.
Ebenfalls identisch zur *total dwell time on erroneous lines* verhalten sich die Zusammenhänge. Es konnten keine statistisch signifikanten Korrelationen für die relevanten Variablen der *allgemeine Programmiererfahrung* (r_{BP}=-.129; p=.467), der *professio-*

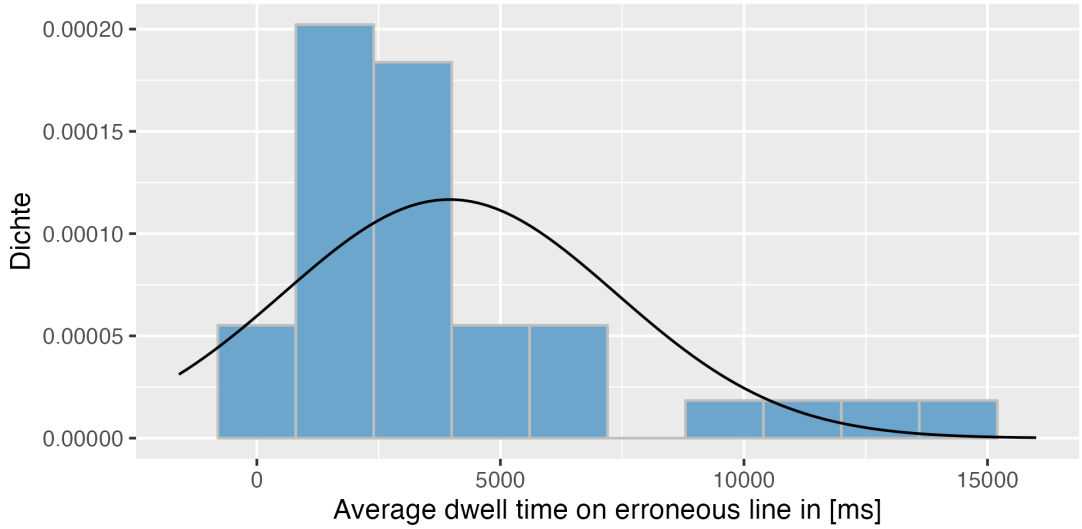

Abbildung 6.14: Verteilung der *average dwell time on erroneous line*

nelle Programmiererfahrung (r_{BP}=-.155; p=.380) oder für die *Anzahl der erzielten Punkte* (r_{BP}=.041; p=.818) beobachtet werden.
Auch der Mann-Whitney-U-Text (U=123; z=.699; r=.120; p=.484) erfasst keine statistisch signifikanten Unterschiede zwischen den Gruppen. Somit kann davon ausgegangen werden, dass sich die Experten (Mdn=3023.583 ms) und Novizen (Mdn=2611.250) nicht signifikant voneinander unterscheiden.

6.3.4 Phasenbasierte Betrachtung der Eye-Tracking-Daten

Nachdem zuvor die allgemeinen Unterschiede zwischen Experten und Novizen hinsichtlich der relevanten Eye-Tracking-Metriken dargestellt wurden, widmet sich der nachfolgende Teil dieses Kapitels der phasenbasierten Betrachtung der Metriken. Auch im Falle dieser Studie wird auf die im vorherigen Kapitel 5 (siehe Absatz
Die Eye-Tracking-Daten wurden erneut auf Basis der *individuellen Bearbeitungszeit* in drei gleich lange Phasen unterteilt. Diese Phasen decken ebenfalls den Grundgedanken der *holistic models of image perception* ab (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000; Sheridan & Reingold, 2017; Swensson, 1980), welche bereits in Kapitel 2.3 beschrieben wurden und deren Annahmen sich auch durch die Vorgängerstudie zogen. Aufgrund der Verwendung von Tobii Hard- und Software wurden die daraus resultierenden Veränderungen in den R-Skripten beachtet.
Identisch zur phasenbasierten Betrachtung der Eye-Tracking-Metriken bei der C-Studie (siehe Absatz 5.3.4), soll auch für C++ untersucht werden, ob sich zwischen den Phasen statistisch signifikante Unterschiede ergeben. Diese Überprüfung greift erneut auf Friedman-Tests zurück, welche durch ergänzende deskriptive Statistik zur besseren Einordnung der Ergebnisse ergänzt werden. Im Rahmen dieser Auswertung wird ebenfalls überprüft, ob sich zwischen den unterschiedlichen Erfahrungsgruppen signifikante Unterschiede ergeben.

Phasenbasierte Betrachtung der *total number of fixations*

Wie bereits bei der zuvor beschriebenen Analyse der grundlegenden Eye-Tracking-Metriken, wird auch bei der phasenbasierten Betrachtung mit der *total number of fixations* begonnen. Die durchgeführten Friedman-Tests weisen auf mehrere statistisch signifikante Unterschiede hin (siehe Tabelle 6.16).

Tabelle 6.16: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of fixations

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=0.272)	Kein Unterschied (p=0.272)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=0.203)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.203)
Novizen	Kein Unterschied (p=1.000)	Unterschied (p=0.036)	Unterschied (p=0.003)
$FT_{(Gesamt)}$: $\chi^2(2)=16.133$; p=0.000 $FT_{(Experten)}$: $\chi^2(2)=6$; p=0.049 $FT_{(Novizen)}$: $\chi^2(2)=13.775$; p=0.001			

Übergreifend über die beiden Erfahrungsstufen zeigt durch deinen Friedman-Test ($FT_{(Gesamt)}$: $\chi^2(2)=16.133$; p=0.000) und anschließende Post-hoc-Tests ein signifikanter Unterschied zwischen der zweiten und dritten Phase (p=0.000). Dieser lässt sich auch in der begleitenden deskriptiven Statistik beobachten (siehe Tabelle 6.17). So weist Phase 2 mit einem Mittelwert von insgesamt 261.735 (SD=182.751; Mdn=198.500) Fixations einen deutlich höheren Wert auf, als Phase 3 mit 217.735 (SD=179.023; Mdn=165.500) Fixations.
Wird die *total number of fixations* nur für die Gruppe der Expertinnen und Experten betrachtet, so ergeben sich keine relevanten Unterschiede zwischen den drei Phasen, obwohl der Friedman-Test statistische Signifikanz erreicht ($FT_{(Experten)}$: $\chi^2(2)=6$; p=.049). Dies kann auch der deskriptiven Statistik in Tabelle 5.19 entnommen werden. Für die Gruppe der Novizinnen und Novizen zeigt der Friedman-Test ($FT_{(Novizen)}$: $\chi^2(2)=13.775$; p=0.001) signifikante Unterschiede an. Die Post-hoc-Tests lokalisieren diese zwischen Phase 1 (M=226.056; SD=159.016; Mdn=178.000) und Phase 3 (M=196.056; SD=193.055; Mdn=115.000) (p=0.036), sowie zwischen Phase 2 (M=241.944; SD=183.805; Mdn=170.500) und Phase 3 (p=0.003).
Die in Tabelle 6.17 dargestellte deskriptive Statistik wird in Abbildung 6.15 als Boxplot dargestellt. Dieser soll die Verteilung der phasenbasierten Betrachtung der *total number of fixations* verdeutlichen.

Phasenbasierte Betrachtung der *average number of fixations*

Die phasenbasierte Betrachtung der *average number of fixations* fördert die gleichen Ergebnisse zutage, wie die im Absatz zuvor beschriebene *total number of fixations* (siehe Absatz 6.3.4). Die Friedman-Tests (siehe 6.18) deuten erneut signifikante Unterschiede zwischen den Phasen an.

Tabelle 6.17: Deskriptive Statistik zur phasenbasierten Betrachtung der total number of fixations

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	238.294	158.653	191.000	58.000	669.000	611.000
	2	261.735	182.751	198.500	60.000	834.000	774.000
	3	217.735	179.023	165.500	27.000	727.000	700.000
Experten	1	252.062	162.283	205.500	58.000	641.000	583.000
	2	284.000	184.899	226.000	60.000	834.000	774.000
	3	242.125	164.546	206.000	27.000	727.000	700.000
Novizen	1	226.056	159.016	178.000	81.000	669.000	588.000
	2	241.944	183.805	170.500	83.000	711.000	628.000
	3	196.056	193.055	115.000	28.000	701.000	673.000

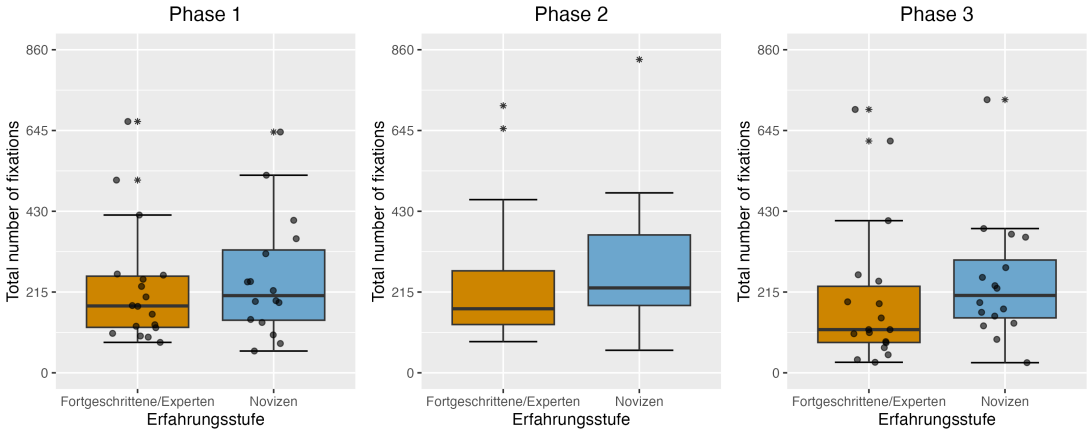

Abbildung 6.15: Betrachtung der phasenbasierten Daten zur *total number of fixations*

Für die gesamte Stichprobe lässt sich ein statistisch signifikanter Unterschied ($FT_{(Gesamt)}$: $\chi^2(2)=16.133$; p=0.000) zwischen der zweiten (M=32.717; SD=22.844; Mdn=24.812) und dritten Phase (M=27.217; SD=22.379; Mdn=20.688) lokalisieren (p=0.000) Ergänzende deskriptive Statistik kann Tabelle 6.19 entnommen werden.
Trotz eines signifikanten Friedman-Tests ($FT_{(Experten)}$: $\chi^2(2)=6$; p=0.049) können die Post-hoc-Tests für die Gruppe der Expertinnen und Experten keine Unterschiede zwischen den einzelnen Phasen feststellen.
Anders verhält sich die Gruppe der Novizinnen und Novizen. Für diese zeigen der Friedman-Test ($FT_{(Novizen)}$: $\chi^2(2)=13.775$; p=0.001) und die begleitenden Post-Hoc-Tests signifikante Unterschiede zwischen den Phasen an. Identisch zur *total number of fixations* befinden sich diese ebenfalls zwischen der ersten (M=28.257; SD=19.877; Mdn=22.250) und dritten Phase (M=24.507; SD=24.132; Mdn=14.375) (p=0.036), sowie zwischen der zweiten (M=30.245; SD=22.976; Mdn=21.312) und dritten Phase (p=0.003).
Auch die deskriptive Statistik zur phasenbasierten Betrachtung der *average number of fixations* wird zur Veranschaulichung als Boxplot dargestellt. Dieser findet sich in Abbildung 6.16.

Tabelle 6.18: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average number of fixations

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=0.272)	Kein Unterschied (p=0.272)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=0.203)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.203)
Novizen	Kein Unterschied (p=1.000)	Unterschied (p=0.036)	Unterschied (p=0.003)

$FT_{(Gesamt)}$: $\chi^2(2)$=16.133; p=0.000
$FT_{(Experten)}$: $\chi^2(2)$=6; p=0.049
$FT_{(Novizen)}$: $\chi^2(2)$=13.775; p=0.001

Tabelle 6.19: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of fixations

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Gesamt	1	29.787	19.832	23.875	7.250	83.25	76.375
	2	32.717	22.844	24.812	7.500	104.25	96.750
	3	27.217	22.379	20.688	3.375	90.875	87.500
Experten	1	31.508	20.285	25.688	7.250	80.125	72.875
	2	35.500	23.112	28.250	7.500	104.25	96.750
	3	30.266	20.568	25.750	3.375	90.875	87.50
Novizen	1	28.257	19.877	22.250	10.125	83.625	73.500
	2	30.245	22.976	21.312	10.375	88.875	78.500
	3	24.507	24.132	14.375	3.500	87.625	84.125

Phasenbasierte Betrachtung der *total fixation duration*

Neben der Anzahl der Fixations wurde auch die Dauer der Fixations in den verschiedenen Phasen betrachtet. Den Anfang bei dieser Analyse macht die *total fixation duration*, welche in Millisekunden erfasst wurde. Für diese Metrik zeigen die Friedman-Tests Unterschiede für alle untersuchten Gruppen an.
Bei der Betrachtung der gesamten Stichprobe zeigen sich signifikante Unterschiede ($FT_{(Gesamt)}$: $\chi^2(2)$=36.059; p=0.000) zwischen Phase 1 (M=133935.500; SD=75920.580; Mdn=109195.000) und Phase 3 (M=112433.700; SD=82985.740; Mdn=85681.000) (p=0.000), sowie zwischen Phase 2 (M=136940.000; SD=77364.470; Mdn=108274.000) (p=0.000).
Die Gruppe der Expertinnen und Experten weist ebenfalls einen signifikanten Friedman-Test auf ($FT_{(Experten)}$: $\chi^2(2)$=19.500; p=0.000). Die begleitenden Post-Hoc-Tests lokalisieren die Unterschiede zwischen Phase 1 (M=138459.400; SD=79584.670; Mdn=121227.500) und Phase 3 (M=122126.100; SD=80489.280; Mdn=117492.500) (p=0.009), sowie zwischen Phase 2 (M=142036.200; SD=77957.830; Mdn=126577.000) und Phase 3 (p=0.000).
Auf Seiten der Novizinnen und Novizen können ebenfalls signifikante Unterschiede zwischen den Phasen beobachtet werden ($T_{(Novizen)}$: $\chi^2(2)$=16.778;

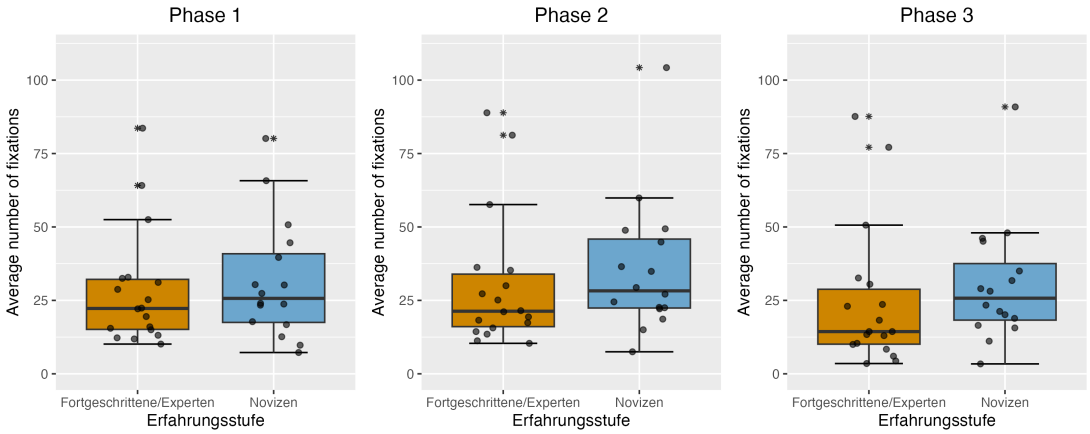

Abbildung 6.16: Betrachtung der phasenbasierten Daten zur *average number of fixations*

Tabelle 6.20: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total fixation duration

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Unterschied (p=0.000)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=1.000)	Unterschied (p=0.009)	Unterschied (p=0.000)
Novizen	Kein Unterschied (p=1.000)	Unterschied (p=0.001)	Unterschied (p=0.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=36.059; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=19.500; p=0.000 $FT_{(Novizen)}$: $\chi^2(2)$=16.778; p=0.000			

p=0.000). Mittels Post-Hoc-Tests können diese zwischen der ersten (M=129914.200; SD=74591.240; Mdn=95871.500) und dritten Phase (M=103818.300; SD=86514.850; Mdn=80500.000) (p=0.001), sowie zwischen der zweiten (M=132411.600; SD=78802.660; Mdn=93729.500) und dritten Phase (p=0.000) identifiziert werden.
Ergänzende deskriptive Statistik zur phasenbasierten Betrachtung der *total fixation duration* findet sich in Tabelle 6.21.
Auch für die *total fixation duration* findet sich eine Veranschaulichung der deskriptiven Statistik in Form eines Boxplots wieder. Dieser wird in Abbildung 6.17 wieder.

Phasenbasierte Betrachtung der *average fixation duration*

Im nächsten Schritt wird die *average fixation duration* hinsichtlich möglicher Unterschiede in den jeweiligen Phasen untersucht. Die durchgeführten Friedman-Tests deuten jedoch lediglich auf einen vorhandenen Unterschied hin (siehe Tabelle 6.22.
Bei der Betrachtung der durchgeführten Friedman-Tests ($FT_{(Gesamt)}$: $\chi^2(2)$=9.235; p=0.001) zeigt sich ein statistisch signifikanter Unterschied für die gesamte Stichprobe. Die Post-Hoc-Tests (p=0.015) identifizieren diesen zwischen Phase 1 (M=612.849; SD=145.831; Mdn=614.157) und Phase 2 (M=572.842; SD=142.393; Mdn=545.704).
Weder für die Expertinnen und Experten ($FT_{(Experten)}$: $\chi^2(2)$=3.875; p=0.144), noch für

Tabelle 6.21: Deskriptive Statistik zur phasenbasierten Betrachtung der total fixation duration in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	133935.500	75920.580	109195.000	43721.000	376974.000	333253.000
	2	136940.900	77364.470	108274.000	47115.000	374458.000	327343.000
	3	112433.700	82985.740	85681.000	12179.000	370713.000	358534.000
Experten	1	138459.400	79584.670	121227.500	43721.000	376974.000	333253.000
	2	142036.200	77957.830	126577.000	58406.000	374458.000	316052.000
	3	122126.100	80489.280	117492.500	35750.000	370713.000	334963.000
Novizen	1	129914.200	74591.240	95871.500	54291.000	342888.000	288597.000
	2	132411.600	78802.660	93729.500	47115.000	343807.000	296692.000
	3	103818.300	86514.850	80500.000	12179.000	357560.000	345381.000

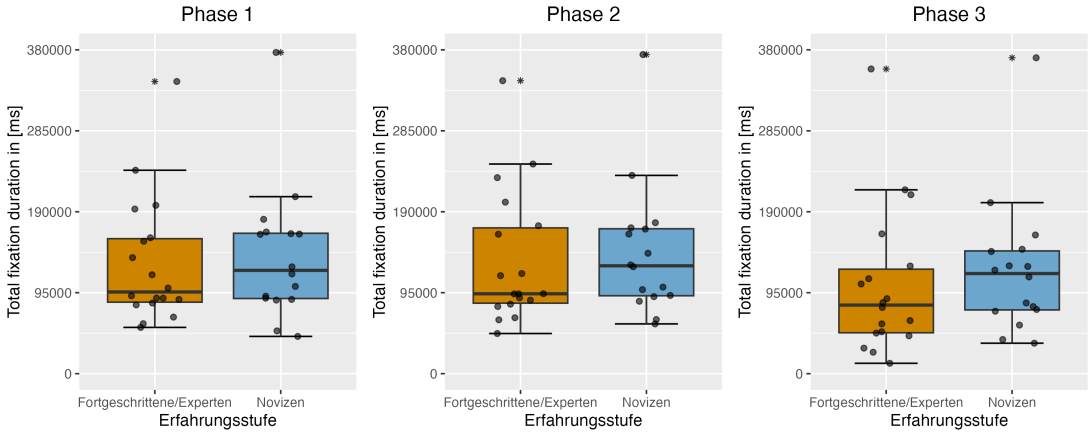

Abbildung 6.17: Betrachtung der phasenbasierten Daten zur *total fixation duration*

die Novizinnen und Novizen ($FT_{(Novizen)}$: $\chi^2(2)=5.444$; p=0.066) können weitere Unterschiede beobachtet werden. Ergänzend dazu finden sich in Tabelle 6.23 ergänzende Informationen zur deskriptiven Statistik hinsichtlich der phasenbasierten Analyse der *average fixation duration*.
Die in Tabelle 6.23 zur Verfügung gestellten Daten werden auch im für die *average fixation duration* grafisch aufbereitet. Der daraus resultierende Boxplot findet sich in Abbildung 6.18.

Phasenbasierte Betrachtung der *fixation rate*

Die *fixation rate* stellt die letzte fixationsbasierte Metrik im Rahmen der phasenbasierten Betrachtung dar. Für sie ergeben sich laut den Friedman-Tests Unterschiede zwischen allen analysierten Phasen (siehe Tabelle 6.24).
Die Betrachtung der gesamten Stichprobe zeigt, dass sich alle Phasen signifikant voneinander unterscheiden ($FT_{(Gesamt)}$: χ^2=68.000; p=0.000). Phase 1 (M=1.615; SD=0.347; Mdn=1.589) unterscheidet sich signifikant von Phase 2 (M=0.748; SD=0.210; Mdn=0.748) (p=0.000) und Phase 3 (M=0.340; SD=0.167; Mdn=0.316) (p=0.000). Weiterhin kann auch ein Unterschied zwischen Phase 2 und 3 (p=0.000) beobachtet wer-

Tabelle 6.22: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average fixation duration

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=0.537)	Unterschied (p=0.015)	Kein Unterschied (p=1.000)
Experten	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.118)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.118)	Kein Unterschied (p=1.000)

$FT_{(Gesamt)}$: $\chi^2(2)$=9.235; p=0.001
$FT_{(Experten)}$: $\chi^2(2)$=3.875; p=0.144
$FT_{(Novizen)}$: $\chi^2(2)$=5.444; p=0.066

Tabelle 6.23: Deskriptive Statistik zur phasenbasierten Betrachtung der average fixation duration in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Gesamt	1	612.849	145.831	614.157	344.331	932.224	587.894
	2	572.842	142.393	545.704	353.892	1058.633	704.741
	3	574.927	184.663	521.099	347.971	1324.074	976.103
Experten	1	599.212	145.915	584.987	344.331	864.647	520.317
	2	547.082	161.817	486.226	393.54	1058.633	665.093
	3	556.814	224.026	501.678	380.091	1324.074	943.983
Novizen	1	624.970	148.881	666.987	376.569	932.224	555.655
	2	595.740	122.779	569.195	353.892	823.926	470.034
	3	591.026	146.041	588.188	347.971	899.357	551.386

den.

Auf Seiten der Expertinnen und Experten kann ein identisches Ergebnis beobachtet werden: Alle drei Phasen unterscheiden sich voneinander ($FT_{(Experten)}$: $\chi^2(2)$=32.000; p=0.000). Die Unterschiede zwischen Phase 1 (M=1.599; SD=0.353; Mdn=1.580) und 2 (M=0.827; SD=0.172; Mdn=0.805) (p=0.028), Phase 1 und 3 (M=0.406; SD=0.136; Mdn=0.388) (p=0.000), sowie zwischen Phase 2 und 3 (p=0.028) erreichen alle statistische Signifikanz.

Das gleiche Bild zeigt sich für die Novizinnen und Novizen. Der Friedman-Test ($FT_{(Novizen)}$: $\chi^2(2)$=26.000; p=0.000) und die begleitenden Post-Hoc-Tests zeigen Unterschiede zwischen allen Phasen auf. So unterscheidet sich Phase 1 (M=1.640; SD=0.352; Mdn=1.615) signifikant von Phase 2 (M=0.679; SD=0.220; Mdn=0.611) (p=0.016) und Phase 3 (M=0.281; SD=0.172; Mdn= 0.247) (p=0.000). Weiterhin erreicht auch der Unterschied zwischen Phase 2 und 3 ein statistisch signifikantes Niveau (p=0.016).

Weitere Informationen zur deskriptiven Statistik können aus Tabelle 6.25 entnommen werden.

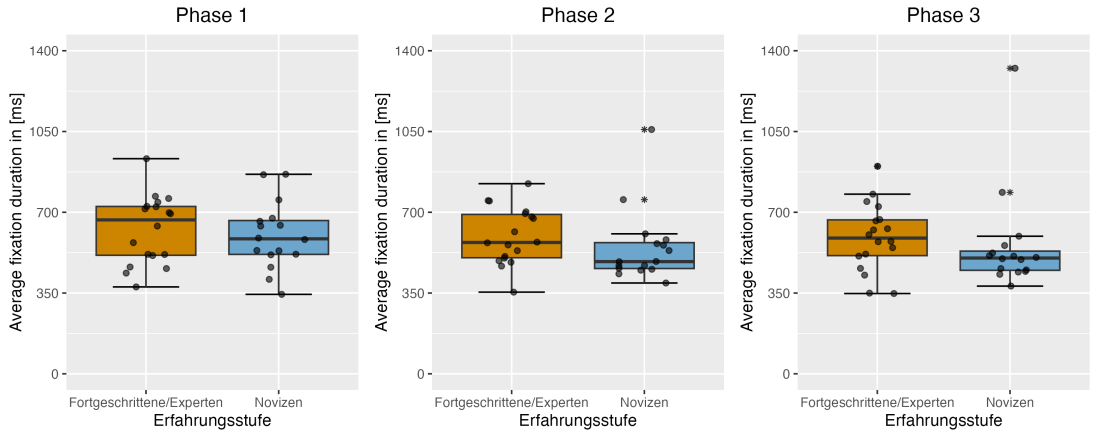

Abbildung 6.18: Betrachtung der phasenbasierten Daten zur *average fixation duration*

Tabelle 6.24: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der fixation rate

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=.000)	Unterschied (p=0.000)	Unterschied (p=.000)
Experten	Unterschied (p=0.028)	Unterschied (p=0.000)	Unterschied (p=0.028)
Novizen	Unterschied (p=0.016)	Unterschied (p=0.000)	Unterschied (p=0.016)
$FT_{(Gesamt)}$: $\chi^2(2)$=68.000; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=32.000; p=0.000 $FT_{(Novizen)}$: $\chi^2(2)$=26.000; p=0.000			

Auch zur deskriptiven Statistik der *fixation rate* wurde ein Boxplot erstellt. Dieser wird in Abbildung 6.19 dargestellt.

Phasenbasierte Betrachtung der *total number of saccades*

Folgend auf die fixationsbasierten Eye-Tracking-Metriken werden nachfolgend die Variablen betrachtet, welche auf Saccaden beruhen. Den Anfang der Analyse macht die *total number of saccades*. Für diese Metrik lassen sich mehrere signifikante Unterschiede zwischen den Phasen beobachten (siehe Tabelle 6.26.
Für die gesamte Stichprobe können über die Friedman-Tests ($FT_{(Gesamt)}$: $\chi^2(2)$=32.548; p=0.000) und die begleitenden Post-Hoc-Tests signifikante Unterschiede zwischen Phase 1 (M=261.029; SD=194.206; Mdn=170.500) und 3 (M=1314.882; SD=869.172; Mdn=1341.500) (p=0.000), sowie zwischen Phase 2 (M=316.059; SD=203.044; Mdn=276.000) und 3 nachgewiesen werden (p=0.000).
Für die Expertinnen und Experten verhält es sich ähnlich. Auch hier liegt ein signifikanter Friedman-Test vor ($FT_{(Experten)}$: $\chi^2(2)$=13.500; p=0.001) und es ergeben sich signifikante Unterschiede zwischen Phase 1 (M=286.000; SD=260.206; Mdn=164.000) und Phase 3 (M=1043.250; SD=909.219; Mdn=911.000) (p=0.009), wie auch zwischen

Tabelle 6.25: Deskriptive Statistik zur phasenbasierten Betrachtung der fixation rate

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	1.615	0.347	1.589	0.908	2.260	1.352
	2	0.748	0.210	0.748	0.379	1.355	0.976
	3	0.340	0.167	0.316	0.075	0.773	0.698
Experten	1	1.599	0.353	1.580	0.908	2.177	1.269
	2	0.827	0.172	0.805	0.512	1.062	0.550
	3	0.406	0.136	0.388	0.126	0.611	0.485
Novizen	1	1.630	0.352	1.615	1.131	2.260	1.129
	2	0.679	0.220	0.611	0.379	1.355	0.976
	3	0.281	0.172	0.247	0.075	0.773	0.698

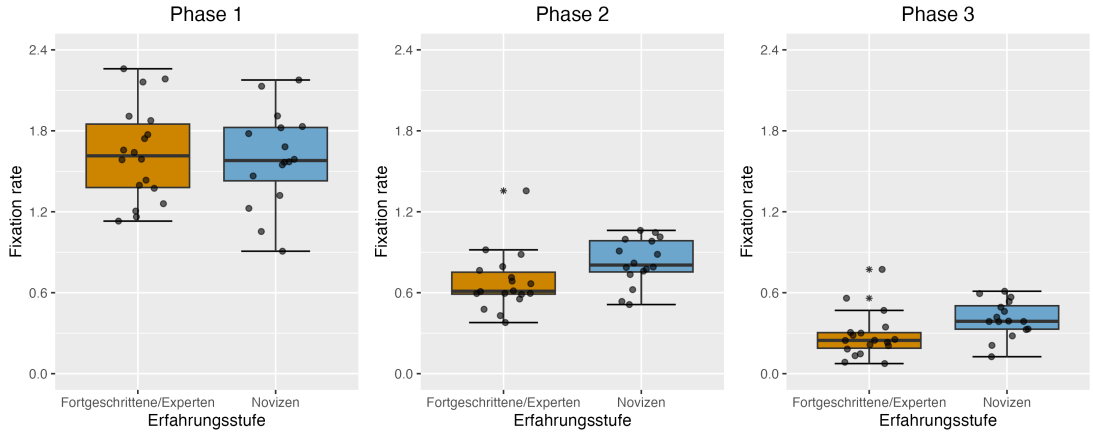

Abbildung 6.19: Betrachtung der phasenbasierten Daten zur *fixation rate*

Phase 2 (M=235.312; SD=145.121; Mdn=211.500) und Phase 3 (p=0.009).
Das gleiche Resultat zeigt sich auch für die Novizinnen und Novizen ($FT_{(Novizen)}$: $\chi^2(2)$=20.310; p=0.000). Hier befinden sich die Unterschiede ebenfalls an den selben Stellen: Phase 1 (M=238.833; SD=111.192; Mdn=206.500) unterscheidet sich signifikant von Phase 3 (M=1556.333; SD=778.387; Mdn=1491.000) (p=0.000) und Phase 2 (M=387.833; SD=223.283; Mdn=371.000) unterscheidet sich ebenfalls von Phase 3 (p=0.003). Die Post-Hoc-Tests zeigen für beide einen signifikanten Wert an.
Weitere Informationen zur deskriptiven Statistik finden sich in Tabelle 6.27.
Auch für die *total number of saccades* werden die deskriptiven Ergebnisse durch einen Boxplot illustriert. Dieser findet sich in Abbildung 6.20 wieder.

Phasenbasierte Betrachtung der *average number of saccades*

Für die *average number of saccades* lassen sich bei der phasenbasierten Betrachtung ähnliche Ergebnisse, wie bei der zuvor beschriebenen *total number of saccades* beobachten. Die durchgeführten Friedman-Tests bestätigen das Vorhandensein von mehreren Unterschieden zwischen den Phasen (siehe Tabelle 6.28).
Der Friedman-Test ($FT_{(Gesamt)}$: $\chi^2(2)$=32.548; p=0.000) und die Post-Hoc-Tests für die

Tabelle 6.26: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of saccades

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Unterschied (p=0.000)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=1.000)	Unterschied (p=0.009)	Unterschied (p=0.009)
Novizen	Kein Unterschied (p=0.680)	Unterschied (p=0.000)	Unterschied (p=0.003)

$FT_{(Gesamt)}$: $\chi^2(2)$=32.548; p=0.000
$FT_{(Experten)}$: $\chi^2(2)$=13.500; p=0.001
$FT_{(Novizen)}$: $\chi^2(2)$=20.310; p=0.000

Tabelle 6.27: Deskriptive Statistik zur phasenbasierten Betrachtung der total number of saccades

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	261.029	194.206	170.500	58.000	840.000	782.000
	2	316.059	203.044	276.000	0.000	913.000	913.000
	3	1314.882	869.172	1341.500	0.000	3502.000	3502.000
Experten	1	286.000	260.206	164.000	58.000	840.000	782.000
	2	235.312	145.121	211.500	36.000	573.000	537.000
	3	1043.250	909.219	911.000	69.000	3502.000	3433.000
Novizen	1	238.833	111.192	206.500	114.000	535.000	421.000
	2	387.833	223.283	371.000	0.000	913.000	913.000
	3	1556.333	778.387	1491.000	0.000	3141.000	3141.000

gesamte Stichprobe zeigen signifikante Unterschiede zwischen der ersten (M=32.629; SD=24.276; Mdn=21.312) und dritten Phase (M=164.360; SD=108.646; Mdn=167.688) (p=0.000), sowie zwischen der zweiten (M=39.507; SD=25.381; Mdn=34.500) und dritten Phase auf (p=0.000).

Das selbe Ergebnis zeigt sich für die Expertinnen und Experten ($FT_{(Experten)}$: $\chi^2(2)$=13.500; p=0.001). Die signifikanten Unterschiede für diese Gruppe befinden sich ebenfalls zwischen Phase 1 (M=35.750; SD=32.526; Mdn=20.500) und 3 (M=130.406; SD=113.652; Mdn=113.875) (p=0.009), sowie zwischen Phase 2 (M=29.414; SD=18.140; Mdn=26.438) und 3 (p=0.009).

Identisch verhalten sich auch die Novizinnen und Novizen ($FT_{(Novizen)}$: $\chi^2(2)$=20.310; p=0.000). Die Post-Hoc-Tests lokalisieren die Unterschiede erneut bei Phase 1 (M=29.854; SD=13.899; Mdn=25.812) und 3 (M=194.542; SD=97.298; Mdn=186.375) (p=0.000), sowie bei Phase 2 (M=48.479; SD=27.910; Mdn=46.375) und 3 (p=0.028).

Weitere deskriptive Statistik zur phasenbasierten Betrachtung der *average number of saccades* findet sich in Tabelle 6.29.

Der in Abbildung 6.21 dargestellte Boxplot veranschaulicht erneut die Ergebnisse zur *average number of saccades*.

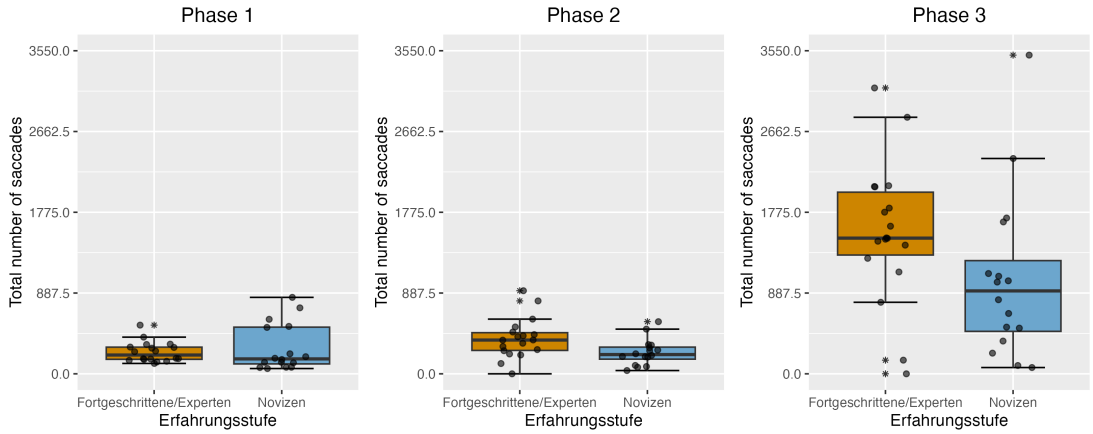

Abbildung 6.20: Betrachtung der phasenbasierten Daten zur *total number of saccades*

Tabelle 6.28: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average number of saccades

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Kein Unterschied (p=1.000)	Unterschied (p=0.000)	Unterschied (p=0.000)
Experten	Kein Unterschied (p=1.000)	Unterschied (p=0.009)	Unterschied (p=0.009)
Novizen	Kein Unterschied (p=0.680)	Unterschied (p=0.000)	Unterschied (p=0.028)
$FT_{(Gesamt)}$: $\chi^2(2)$=32.548; p=0.000 $FT_{(Experten)}$: $\chi^2(2)$=13.500; p=0.001 $FT_{(Novizen)}$: $\chi^2(2)$=20.310; p=0.000			

Phasenbasierte Betrachtung der *total number of visits on erroneous lines*

Auf die saccadenbasierten Metriken folgend, werden im Anschluss die AOI-basierten Metriken betrachtet. Es wird mit der *total number of visits on erroneous lines* begonnen. Die Friedman-Tests (siehe Tabelle 6.30 zeigen in diesem Fall ein gemischtes Bild.
Übergreifend über die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)$=14.970; p=0.001) lassen sich signifikante Unterschiede beobachten. Diese befinden sich zwischen der ersten (M=9.765; SD=11.586; Mdn=4.000) und zweiten Phase (M=18.735; SD=19.551; Mdn=11.500) (p=0.001), sowie zwischen der zweiten und dritten Phase (M=13.912; SD=18.587; Mdn=7.000) (p=0.092). Obwohl seitens des Post-Hoc-Tests für den Unterschied zwischen Phase 2 und 3 ein Unterschied gemeldet wird, erreicht dieser mit einem Wert von p=0.092 keine statistische Signifikanz.
Für die Gruppe der Expertinnen und Experten ($FT_{(Experten)}$: $\chi^2(2)$=15.129; p=0.000) zeigt sich lediglich ein Unterschied. Dieser kann zwischen Phase 1 (M=7.500; SD=8.246; Mdn=3.500) und Phase 2 (M=21.688; SD=18.205; Mdn=16.000) beobachtet werden (p=0.000).
Bei der Betrachtung der Novizinnen und Novizen zeigen sich keine signifikanten Unterschiede ($FT_{(Novizen)}$: $\chi^2(2)$=2.771; p=0.250).

Tabelle 6.29: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of saccades

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	32.629	24.276	21.312	7.250	105.000	97.750
	2	39.507	25.381	34.500	0.000	114.125	114.125
	3	164.360	108.646	167.688	0.000	437.750	437.750
Experten	1	35.750	32.526	20.500	7.250	105.000	97.750
	2	29.414	18.140	26.438	4.500	71.625	67.125
	3	130.406	113.652	113.875	8.625	437.75	429.125
Novizen	1	29.854	13.899	25.812	14.250	66.875	52.625
	2	48.479	27.910	46.375	0.000	114.125	114.125
	3	194.542	97.298	186.375	0.000	392.625	392.625

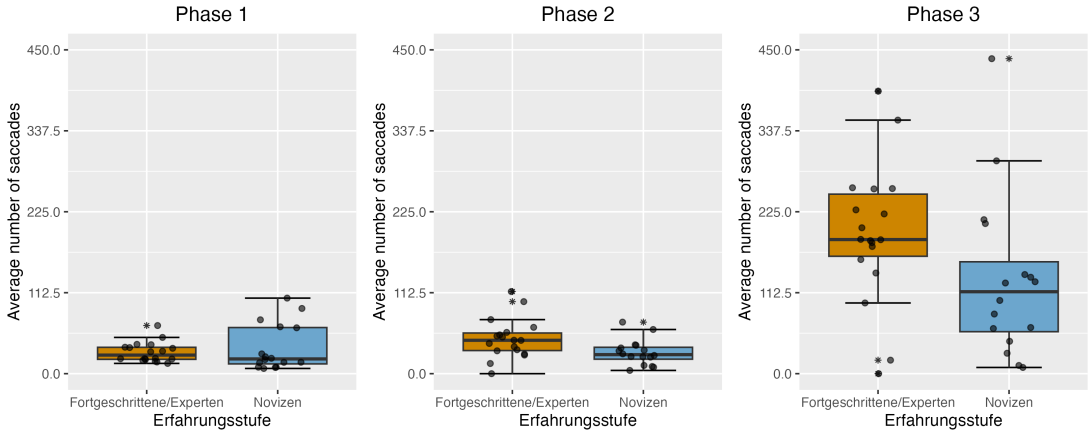

Abbildung 6.21: Betrachtung der phasenbasierten Daten zur *average number of saccades*

Weitere Informationen zur deskriptiven Statistik der phasenbasierten Analyse der *total number of visits on erroneous lines* kann Tabelle 6.31 entnommen werden.
Erneut werden die Ergebnisse grafisch veranschaulicht. Der dazugehörige Boxplot zur phasenbasierten Betrachtung der *total number of visits on erroneous lines* findet sich in Abbildung 6.22.

Phasenbasierte Betrachtung der *average number of visits on erroneous lines*

Die phasenbasierte Betrachtung der *average number of visits on erroneous lines* verhält sich identisch zur zuvor beschriebenen *total number of visits*. Die Friedman-Tests identifizieren erneut Unterschiede zwischen den Phasen (siehe 6.32).
Für die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)=14.970$; p=0.001) zeigen sich erneut zwei Unterschied zwischen der ersten (M=0.697; SD=0.828; Mdn=0.286) und zweiten (M=1.338; SD=1.397; Mdn=0.821) (p=0.000), sowie zwischen der zweiten und dritten (M=0.994; SD=1.328; Mdn=0.500) Phase (p=0.092). Letzterer wird zwar von den Post-Hoc-Tests erfasst, erreicht jedoch keine statistische Signifikanz.
Bei den Expertinnen und Experten ($FT_{(Experten)}$: $\chi^2(2)=15.129$; p=0.000) kann ebenfalls zwischen der ersten (M=0.536; SD=0.589; SD=0.250) und zweiten Phase (M=1.549;

Tabelle 6.30: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total number of visits on erroneous lines

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.001)	Kein Unterschied (p=1.000)	Unterschied (p=0.092)
Experten	Unterschied (p=0.000)	Kein Unterschied (p=0.798)	Kein Unterschied (p=0.129)
Novizen	Kein Unterschied (p=0.680)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)$=14.970; p=0.001 $FT_{(Experten)}$: $\chi^2(2)$=15.129; p=0.000 $FT_{(Novizen)}$: $\chi^2(2)$=2.771; p=0.250			

Tabelle 6.31: Deskriptive Statistik zur phasenbasierten Betrachtung der total number of visits on erroneous lines

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	9.765	11.586	4.000	0.000	50.000	50.000
	2	18.735	19.551	11.500	1.000	81.000	80.000
	3	13.912	18.587	7.000	0.000	75.000	75.000
Experten	1	7.500	8.246	3.500	0.000	33.000	33.000
	2	21.688	18.205	16.000	2.000	72.000	70.000
	3	13.438	17.836	7.500	0.000	69.000	69.000
Novizen	1	11.778	13.838	4.000	0.000	50.000	50.000
	2	16.111	20.835	8.000	1.000	81.000	80.000
	3	14.333	19.736	5.500	0.000	75.000	75.000

SD=1.300; Mdn=1.143) ein Unterschied ermittelt werden. Dieser erreicht ein signifikantes Niveau (p=0.000).
Keine Unterschiede lassen sich für die Novizinnen und Novizen beobachten. Der durchgeführte Friedman-Test kann keine Signifikanz feststellen ($FT_{(Novizen)}$: $\chi^2(2)$=2.771; p=0.250).
Ergänzende Informationen zur deskriptiven Statistik werden in Tabelle 6.33 aufgeführt.
Der in Abbildung 6.23 dargestellte Boxplot veranschaulicht die gewonnenen Erkenntnisse.

Phasenbasierte Betrachtung der *total dwell time on erroneous lines*

Die Analyse der phasenbasierten Unterschiede hinsichtlich der *total dwell time on erroneous lines in [ms]* zeigt ebenfalls Unterschiede zwischen den Phasen und den Erfahrungsstufen auf (siehe Tabelle 6.34.
Für die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)$=7.471; p=0.024) ergibt sich laut des Post-Hoc-Tests ein Unterschied zwischen Phase 1 (M=6.330.471; SD=6326.291;

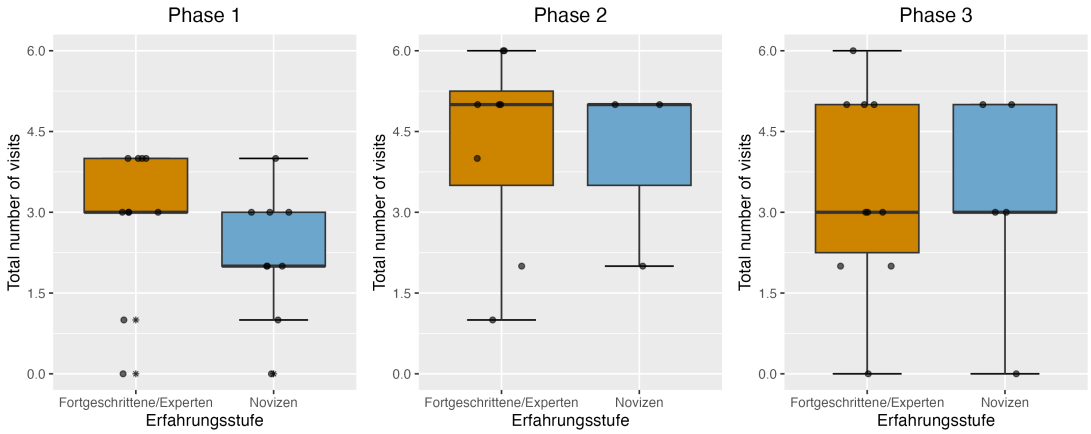

Abbildung 6.22: Betrachtung der phasenbasierten Daten zur *total number of visits on erroneous lines*

Tabelle 6.32: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der *average number of visits on erroneous lines*

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.001)	Kein Unterschied (p=1.000)	Unterschied (p=0.092)
Experten	Unterschied (p=0.000)	Kein Unterschied (p=0.798)	Kein Unterschied (p=0.129)
Novizen	Kein Unterschied (p=0.680)	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)
$FT_{(Gesamt)}$: $\chi^2(2)=14.970$; p=0.001 $FT_{(Experten)}$: $\chi^2(2)=15.129$; p=0.000 $FT_{(Novizen)}$: $\chi^2(2)=2.771$; p=0.250			

Mdn=5121.500) und 2 (M=9557.941; SD=7873.645; Mdn=6657.500). Allerdings erreicht dieser keine statistische Signifikanz (p=0.092).
Auf Seiten der Experten und Expertinnen ($FT_{(Experten)}$: $\chi^2(2)=7.125$; p=0.028) kann ebenfalls ein Unterschied beobachtet werden. Dieser befindet sich zwischen Phase 1 (M=4795.938; SD=4543.511; Mdn=2997.500 und Phase 2 (M=11106.250; SD=8507.450; Mdn=10182.000) (p=0.048).
Die Ergebnisse für die Novizinnen und Novizen ($FT_{(Novizen)}$: $\chi^2(2)=4.778$; p=0.092) zeigen keine signifikanten Unterschiede auf.
Wie auch bei den zuvor analysierten Metriken, wird auch für die *total dwell time on erroneous lines* die ergänzende deskriptive Statistik in Tabelle 6.35 aufgeführt.
Die erlangten Ergebnisse werden ebenso in einem Boxplot zusammengefasst (siehe Abbildung 6.24).

Phasenbasierte Betrachtung der *average dwell time on erroneous line*

Die letzte Metrik, welche im Falle der phasenbasierten Betrachtung analysiert wird ist die *average dwell time on erroneous lines in [ms]*. Für diese zeigt sich ein identisches Bild zur zuvor beschriebenen *total dwell time*. Auch in diesem Fall lassen sich Unterschiede

Tabelle 6.33: Deskriptive Statistik zur phasenbasierten Betrachtung der average number of visits on erroneous lines

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	0.697	0.828	0.286	0.000	3.571	3.571
	2	1.338	1.397	0.821	0.071	5.786	5.714
	3	0.994	1.328	0.500	0.000	5.357	5.357
Experten	1	0.536	0.589	0.250	0.000	2.357	2.357
	2	1.549	1.300	1.143	0.143	5.143	5.000
	3	0.960	1.274	0.536	0.000	4.929	4.929
Novizen	1	0.841	0.988	0.286	0.000	3.571	3.571
	2	1.151	1.488	0.571	0.071	5.786	5.714
	3	1.024	1.410	0.393	0.000	5.357	5.357

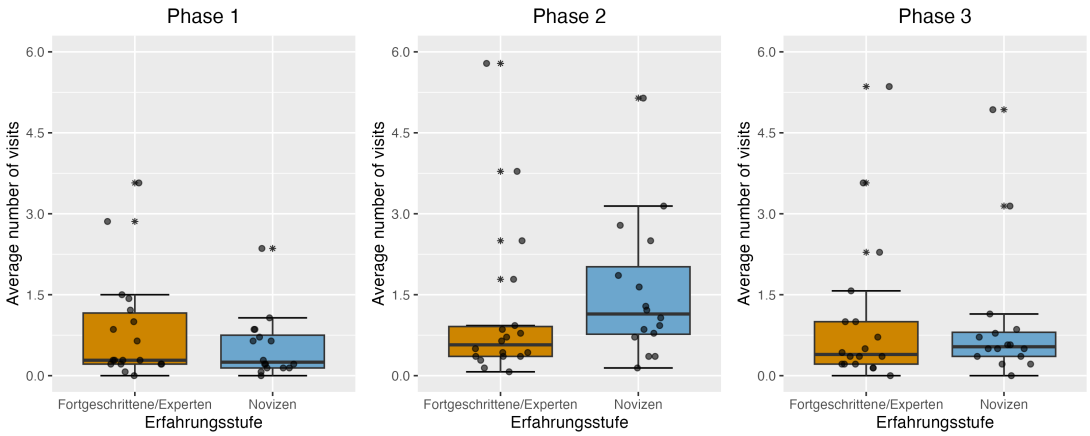

Abbildung 6.23: Betrachtung der phasenbasierten Daten zur *average number of visits on erroneous lines*

beobachten (siehe Tabelle 6.34).
Für die gesamte Stichprobe ($FT_{(Gesamt)}$: $\chi^2(2)=7.471$; p=0.024) ergibt sich ein Unterschied zwischen der ersten (M=452.176; SD=451.878; Mdn=451.878) und zweiten Phase (M=674.853; SD=562.403; Mdn=475.536), welcher jedoch keine statistische Signifikanz erreicht (p=0.092).
Die Betrachtung der Expertinnen und Experten ($FT_{(Experten)}$: $\chi^2(2)=7.125$; p=0.028) zeigt, dass sich für die *average dwell time on erroneous lines* ein signifikanter Unterschied (p=0.048) zwischen der ersten (M=342.567; SD=324.537; Mdn=214.107) und zweiten Phase (M=793.304; SD=607.675; Mdn=727.286) ergibt.
Im Falle der Novizinnen und Novizen ($FT_{(Novizen)}$: $\chi^2(2)=4.778$; p=0.092) können bei dieser Metrik keine Unterschiede festgestellt werden.
Wie bei den vorher beschriebenen Metriken, werden auch die weiteren deskriptiven Ergebnisse tabellarisch aufgeführt. Diese können Tabelle 6.37 entnommen werden.
Auch für die *average dwell time on erroneous lines* wurde ein Boxplot erstellt. Dieser findet sich in Abbildung 6.25 wieder.

Tabelle 6.34: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der total dwell time on erroneous lines

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.092)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.127)
Experten	Unterschied (p=0.048)	Kein Unterschied (p=0.670)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.182)
$FT_{(Gesamt)}$: $\chi^2(2)=7.471$; p=0.024 $FT_{(Experten)}$: $\chi^2(2)=7.125$; p=0.028 $FT_{(Novizen)}$: $\chi^2(2)=4.778$; p=0.092			

Tabelle 6.35: Deskriptive Statistik zur phasenbasierten Betrachtung der total dwell time on erroneous lines in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	6330.471	6326.291	5121.500	0.000	30569.000	30569.000
	2	9447.941	7873.645	6657.500	725.000	34842.000	34117.000
	3	7961.412	9370.203	3702.500	0.000	33934.000	33934.000
Experten	1	4795.938	4543.511	2997.500	297.000	12993.000	12696.000
	2	11106.250	8507.450	10182.000	725.000	34842.000	34117.000
	3	8888.500	10230.470	3960.000	1855.000	33934.000	32079.000
Novizen	1	7694.500	7435.626	6737.000	0.000	30569.000	30569.000
	2	7973.889	7182.699	4887.000	843.000	25723.000	24880.000
	3	7137.333	8749.826	3438.500	0.000	32358.000	32358.000

6.4 Diskussion

Die erlangten Ergebnisse sollen nachfolgend diskutiert und mit den Erkenntnissen aus anderen Studien verglichen werden. Die Diskussion soll dabei ebenfalls in zwei Absätze unterteilt werden: Es wird mit den allgemeinen Eye-Tracking-Metriken begonnen. Diesen folgt die Diskussion der phasenbasierten Metriken.

6.4.1 Diskussion der allgemeinen Eye-Tracking-Metriken

Die zur Analyse der allgemeinen Eye-Tracking-Metriken genutzte Methodik förderte mehrere diskussionswürdige Ergebnisse zu Tage. Diese sollen nachfolgend behandelt werden.

Entgegen der Annahmen der verschiedenen *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980) zeigten sich weder für die *total* und *average number of fixations*, noch für die *total* und *average fixation duration* signifikante Ergebnisse. Es konnten lediglich marginale Unterschiede für die genannten Metriken zwischen den

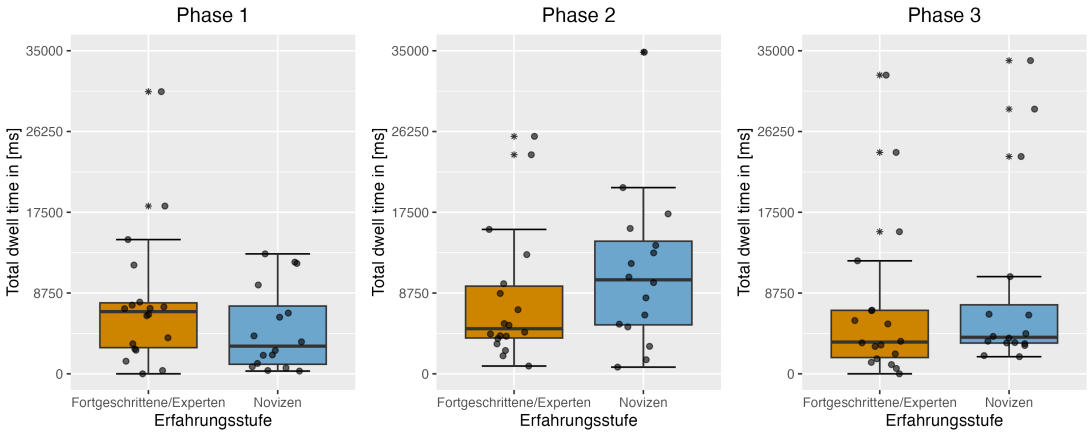

Abbildung 6.24: Betrachtung der phasenbasierten Daten zur *total dwell time on erroneous lines*

Tabelle 6.36: Friedman-Tests zur Erkennung von phasenbasierten Unterschieden bei der average dwell time on erroneous lines

Gruppe	Phase 1 vs. Phase 2	Phase 1 vs. Phase 3	Phase 2 vs. Phase 3
Gesamt	Unterschied (p=0.092)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.127)
Experten	Unterschied (p=0.048)	Kein Unterschied (p=0.670)	Kein Unterschied (p=1.000)
Novizen	Kein Unterschied (p=1.000)	Kein Unterschied (p=1.000)	Kein Unterschied (p=0.182)

$FT_{(Gesamt)}$: $\chi^2(2)=7.471$; p=0.024
$FT_{(Experten)}$: $\chi^2(2)=7.125$; p=0.028
$FT_{(Novizen)}$: $\chi^2(2)=4.778$; p=0.092

beiden Erfahrungsstufen beobachtet werden. Auch entsprechende Korrelationen in Bezug auf die *allgemeine* und *professionelle Programmiererfahrung*, sowie die *Anzahl der im Experiment erzielten Punkte* konnten nur schwache, nicht signifikante Ergebnisse nachweisen. Gleiches gilt für Unterschiede zwischen den Erfahrungsstufen. Interessant ist hingegen, dass sich die fixationsbasierten Metriken gegensätzlich zu den Annahmen von Sheridan und Reingold (2017) verhalten. Aus Sicht der *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980), sowie den Theorien zur visuellen Expertise (Gegenfurtner et al., 2011; Gegenfurtner, 2020; Kok, 2016) liefern diese Metriken nur bedingt verwertbare Ergebnisse.
Anders verhält sich die *fixation rate*. Diese zeigt signifikante Unterschiede zwischen den beiden Gruppen auf. Die Gruppe der Expertinnen und Experten (M=1.614; SD=0.317; Mdn=1.664) hat im Vergleich zu den Novizinnen und Novizen (M=1.407; SD=0.385; Mdn=1.226) einen signifikant höheren Wert. Ebenso zeigt sich für die *fixation rate* ein Zusammenhang mit der Anzahl der erzielten Punkte. Diese Ergebnisse sind identisch zu den von Sheridan und Reingold (2017) formulierten Annahmen zu den *holistic models of image perception*. Im weiteren Sinne deutet die erhöhte *fixati-*

Tabelle 6.37: Deskriptive Statistik zur phasenbasierten Betrachtung der average dwell time on erroneous lines in [ms]

Gruppe	Phase	Mittelwert	SD	Mdn	Min	Max	Range
Komplett	1	452.176	451.878	451.878	0.000	2183.500	2183.500
	2	674.853	562.403	475.536	51.786	2488.714	2436.929
	3	568.672	669.300	264.464	0.000	2423.857	2423.857
Experten	1	342.567	324.537	214.107	21.214	928.071	906.857
	2	793.304	607.675	727.286	51.786	2488.714	2436.929
	3	634.893	730.748	282.857	132.500	2423.857	2291.357
Novizen	1	549.607	531.116	481.214	0.000	2183.500	2183.500
	2	569.563	513.050	349.071	60.214	1837.357	1777.143
	3	509.810	624.988	245.607	0.000	2311.286	2311.286

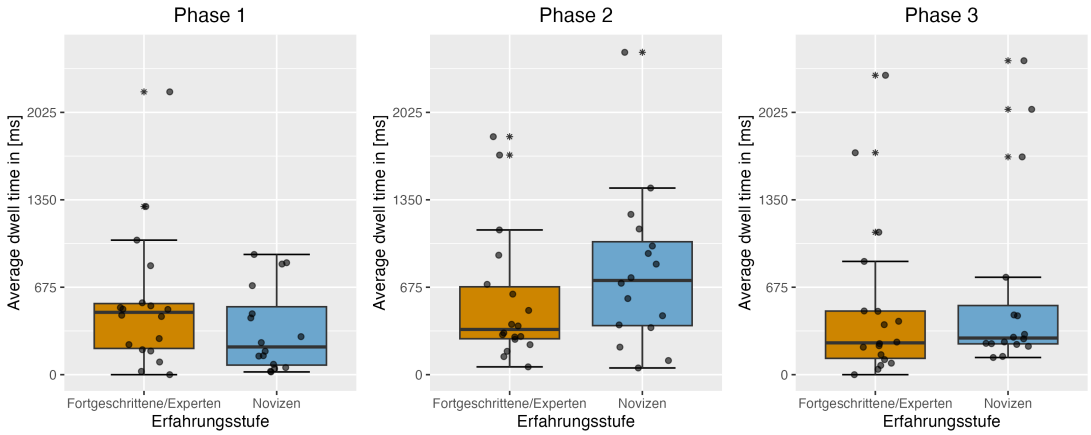

Abbildung 6.25: Betrachtung der phasenbasierten Daten zur *average dwell time on erroneous lines*

on rate auch auf die Durchführung eines Scans oder einer globalen Betrachtung der präsentierten Codebeispiele hin (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Peachock et al., 2017; Sharif et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980; Uwano et al., 2006).
Hinsichtlich der allgemeinen Betrachtung der *total* und *average number of saccades* ergeben sich Unterschiede zwischen den beiden Erfahrungsstufen. Die Mann-Whitney-U-Tests belegen signifikante Unterschiede zwischen den beiden Gruppen. Die in beiden Fällen niedrigere Anzahl von Saccaden für die Expertinnen und Experten spricht dafür, dass sich diese Versuchspersonen effizienter durch die präsentierten Quellcodes navigieren. Auch die nicht signifikanten Korrelationskoeffizienten deuten darauf hin, dass mit zunehmender Erfahrung weniger Saccaden zur Betrachtung der Stimuli benötigt werden. Daraus lässt sich für die Expertinnen und Experten schlussfolgern, dass bestimmte Techniken angewandt werden um sich den Code oder zumindest dessen Struktur einzuprägen. Auch hier lässt sich auf die Effekte von visueller Expertise schließen. Sheridan und Reingold (2017) konnten in ihrer Metastudie ähnliche Ergebnisse für die Betrachtung von medizinischen Stimuli beobachten. Mit steigender Expertise sinkt die Anzahl der benötigten Saccaden ab. Sie (Sheridan & Reingold,

2017) stützen ihre Erkenntnis auf eine Reihe von analysierten Studien (Alzubaidi, Black, Patel & Panchanathan, 2009; Assaf et al., 2016; Donovan & Litchfield, 2013; Krupinski, Graham & Weinstein, 2013). Auch aus Sicht des *global-focal search models* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010), sowie auf Basis von Swenssons (1980) *two-stage detection models*, macht diese Beobachtung Sinn. Im Falle des *global-focal search models* wird durch die globale Betrachtung der Stimuli ein Überblick über diesen gewonnen. Basierend auf diesem erfolgt die fokale Betrachtung von relevanten Arealen. Swensson (1980) führt in seinem Model ergänzend dazu noch die Rolle von Vorwissen auf, welches eine Filterfunktion übernimmt und die Blicke der erfahreneren Versuchspersonen lenkt. Weiterhin ließe sich auch mit dem von Kundel et al. (2007) beschriebenen Ansatz des *holistic mode vs. search to find* argumentieren. Die Experten und Expertinnen nuten den *holistic mode*, wissen aufgrund ihrer Erfahrung bereits mehr über den Aufbau des Codes, gewinnen aus der initialen Betrachtung mehr relevante Informationen und konzentrieren sich in der Folge verstärkt auf Auffälligkeiten. Die Novizinnen und Novizen können im Gegensatz dazu lediglich auf den langsameren und weniger effizienten Ansatz des *search to find* zurückgreifen, der mit einer wiederholten Betrachtung der Stimuli einhergeht und in einer höheren Anzahl an Saccaden resultiert. Insgesamt verhält sich die allgemeine Betrachtung der *total* und *average number of saccades* sehr ähnlich zu den genannten Annahmen der *holistic models of image perception* und belegt das Vorhandensein von erfahrungsbedingten Unterschieden bei Code Reviews in der Programmiersprache C++.

Auf Seiten der AOI-basierten Metriken zeigten sich für die *total* und *average number of visits on erroneous lines* keine nennenswerten Auffälligkeiten. Basierend auf den Annahmen aus der Radiologie (Gegenfurtner et al., 2011; Kok, 2016; Reingold & Sheridan, 2012; Sheridan & Reingold, 2017), hätte sich für die Gruppe der Expertinnen und Experten eine höhere Anzahl an *visits* ergeben sollen. Im Falle der allgemeinen Betrachtung der Metriken liegen jedoch die Mittelwerte beider Gruppen verhältnismäßig nahe zusammen. Die Mediane für die *total* und *average number of visits on erroneous lines* driften jedoch weiter auseinander und zeigen für die erfahreneren Probanden einen größeren Wert an. Dies würde implizieren, dass sich Expertinnen und Experten die fehlerhaften Codezeilen häufiger ansehen, als die Novizinnen und Novizen. Auch die *total* und *average dwell time on erroneous lines* zeigen ein ähnliches Ergebnis. Die Expertinnen und Experten betrachten die fehlerhaften Zeilen zwar länger, jedoch ist der Unterschied zu den weniger erfahrenen Versuchspersonen lediglich marginal und erreicht keine statistische Signifikanz. Basierend auf der Betrachtung dieser Metriken fällt es schwer, ein eindeutiges Fazit zu deren Rolle im Review zu sagen. Tendenziell scheinen sich sowohl die *number of visits*, als auch die *dwell time on erroneous lines* im Sinne der *holistic models of image perception* (Sheridan & Reingold, 2017) zu verhalten. Jedoch liefern die erzielten Ergebnisse nicht genug gesicherte Erkenntnisse um beispielsweise die Annahmen des *global-focal search models* zu verifizieren. Im weiteren Sinne wäre es für beide Metriken empfehlenswert, diese zu korrekten

Codezeilen ins Verhältnis zu setzen, was aus Sicht der Datenauswertung mit einem verhältnismäßig hohem Mehraufwand verbunden wäre.

6.4.2 Diskussion der phasenbasierten Eye-Tracking-Metriken

Die Analyse der phasenbasierten Eye-Tracking-Metriken förderte mehrere diskussionswürdige Ergebnisse zu Tage. Die auf den Friedman-Tests beruhende Vorgehensweise konnte zeigen, dass sich die betrachteten Gruppen der Expertinnen und Experten, sowie die Novizinnen und Novizen voneinander unterscheiden und sich die analysierten Eye-Tracking-Metriken im Verlauf des Experiments verändern.
Aus Sicht der im Theorieteil besprochenen Annahmen zur visuellen Expertise und den ebenfalls dargelegten *holistic models of image perception* (siehe 2.3) ergeben sich mehrere Auffälligkeiten. Besonders hervortraten diese bei der phasenbasierten Betrachtung der *fixation rate*, der *total* und *average number of saccades*, der *total* und *average number of visits on erroneous lines*, sowie bei der *total* und *average dwell time on erroneous lines*.
Bei der *fixation rate* fiel auf, dass diese übergreifend über beide Gruppen im Verlauf des Experiments abfiel. Expertinnen und Experten begannen mit einem Mittelwert von 1.599 Fixations/sec (SD=0.353, Mdn=1.580), welcher in der zweiten Phase auf 0.827 Fixations/sec (SD=0.172; Mdn=0.805) und in der dritten Phase weiter auf 0.406 Fixations/sec (SD=0.136; Mdn=0.388) absank. Für die Novizinnen und Novizen ließ sich ein ähnlicher Trend beobachten. Diese begannen in der ersten Phase mit einem Mittelwert von 1.630 Fixations/sec (SD=0.352; Mdn=1.615), welcher sich in der zweiten Phase auf 0.679 (SD=0.220; Mdn=0.611) Fixations/sec reduzierte und in der dritten Phase nur noch 0.281 Fixations/sec (SD=0.172; Mdn=0.247) betrug. Dieses kontinuierliche Abfallen der *fixation rate* im Verlauf des Experiments deutet darauf hin, dass seitens der Probanden zu Beginn der Betrachtung der Codebeispiele eine Art *scan* bzw. eine globale Betrachtung durchgeführt wird (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sharif et al., 2012; Sheridan & Reingold, 2017; Swensson, 1980; Uwano et al., 2006). Diese eher oberflächliche Betrachtung scheint den Versuchspersonen dazu zu dienen, sich einen Überblick zu verschaffen und wandelt sich im Laufe des Experiments zu einer genaueren Betrachtung, welche aus Sicht des *global-focal search models* als fokale Betrachtung bezeichnet werden kann, in deren Rahmen auch Details und Anomalien näher betrachtet werden (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010). Basierend auf früheren Studien (Sharif et al., 2012; Uwano et al., 2006) und den *holistic models of image perception* deutet die *fixation rate* im Falle dieser Studie auf das Vorhandensein eines *scans*, sowie auf globale und fokale Phasen bei der Durchführung des Code Reviews hin.
Ähnliches zeigt sich auch bei der Betrachtung der *total* und *average number of saccades*. Bei diesen Metriken lässt sich sowohl für die Expertinnen und Experten, als auch für die Novizinnen und Novizen ein ausgeprägter Unterschied zur dritten Phase beobachten. In dieser letzten Phase des Reviews nimmt die Anzahl an Saccaden stark

zu. Erneut auf Basis der verschiedenen *holistic models of image perception* argumentierend, lässt sich festhalten, dass die gesteigerte Anzahl von Saccaden in der dritten Pahse als ein Anzeichen für genaueres Lesen zu werten ist (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980). Die anfangs relativ kleine Anzahl an Saccaden kann in diesem Fall eher so gesehen werden, dass mit diesen ein Überblick über das jeweilige Codebeispiel aufgebaut wird. Dies kann mit wenigen Saccaden erfolgen. Genaueres lesen, beispielsweise bei der Analyse von Auffälligkeiten oder der Suche nach Fehlern erfordert eine fokalere Vorgehensweise und würde mit einer gesteigerten Anzahl von Saccden einhergehen (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010). Dieser Trend zeichnet sich so auch in den analysierten Daten ab. Daraus folgernd, kann davon ausgegangen werden, dass sich die *total* und *average number of saccades* im Falle dieser C++-Studie ebenfalls ähnlich zu den Annahmen der *holistic models of image perception* verhält.
Ein etwas anderes Bild konnte hinsichtlich der AOI-basierten Metriken beobachtet werden. Sowohl für die *total* und *average number of visits on erroneous lines*, als auch für die *total* und *average dwell time on erroneous lines* zeigen sich Unterschiede zwischen den analysierten Phasen. Diese machen sich jedoch nur in der Gruppe der Expertinnen und Experten bemerkbar. Für die *total* (M=7.500; SD=8.246; Mdn=3.500) und *average number of visits on erroneous lines* (M=0.536; SD=0.589; Mdn=0.250) fällt auf, dass diese in der ersten Phase am wenigsten betrachtet wird. Im Verlauf des Reviews erhalten diese jedoch deutlich mehr Aufmerksamkeit. Dies kann im Sinne der holistischen Modelle der Bildbetrachtung (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980) und auf Basis ähnlicher Studien im Software Engineering (Peachock et al., 2017; Sharif et al., 2012; Uwano et al., 2006) ebenfalls als ein Hinweis auf einen initialen Scan gewertet werden. Dieser scheint jedoch für die fortgeschrittenen Versuchspersonen effizienter auszufallen, da deren Gesamtleistung im Experiment mit einer besseren Fehlererkennung einhergeht. Für die *average* und *total dwell time on erroneous lines* zeigt sich in der Datenauswertung ein vergleichbares Ergebnis: Die Expertinnen und Experten lassen den fehlerhaften Codezeilen in der ersten Phase am wenigsten Betrachtungszeit zukommen. Diese erreicht in der zweiten Phase ihren höchsten Wert und weist auch in der dritten Phase einen erhöhten, jedoch nicht signifikanten (p=0.670) Wert auf (siehe die Tabellen 6.34 und 6.35, sowie 6.36 und 6.37). Für die Novizinnen und Novizen in diesem Experiment ergibt sich übergreifend über alle Phasen eher eine Gleichverteilung der *dwell time* (siehe die Tabellen 6.35 und 6.37). Keine der analysierten Phasen für diese Gruppe einen signifikanten Unterschied auf (siehe die Tabellen 6.34 und 6.36). Argumentiert man auch in diesem Falle auf Baiss der *holistic models of image perception*, so zeigen sich Ähnlichkeiten mit dem *global-focal search model* (Nodine & Mello-Thoms, 2000, 2010). Seitens der Experten deutet sich ein initialer Scan an, welcher dazu dient mögliche Fehler oder Anomalien zu identifizieren, welche im weiteren Verlauf des Reviews genauer betrachtet werden. Ähnliche

Vorgehensweisen finden sich auch in der Radiologie (Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Kok, 2016; Sheridan & Reingold, 2017), welche das gleiche Ziel verfolgen: Fehler und Auffälligkeiten zu identifizieren und im nächsten Schritt genauer zu analysieren. Ergänzend dazu liefern die AOI-basierten Metriken auch Hinweise darauf, dass auf Expertinnen und Experten die *information-reduction hypothesis* zutrifft (Canham & Hegarty, 2010; Hmelo-Silver, 2004; Jarodzka et al., 2010; Gegenfurtner et al., 2011; Haider & Frensch, 1996, 1999; Kok, 2016). Die erfahreneren Versuchspersonen verfügen über ein besseres Verständnis über den Aufbau von Quellcodes und sind dazu in der Lage, vulnerable Stellen zu identifizieren und lassen diese ein erhöhtes Maß an Aufmerksamkeit zukommen.
Es konnten weiterhin auch Auffälligkeiten bezüglich der *total* und *average number of fixations* ermittelt werden. Im Falle dieser Metriken ist jedoch vorwiegend die Gruppe der Novizinnen und Novizen betroffen. Für diese zeigt sich in beiden Fällen, dass die Anzahl der Fixationen in der dritten Phase des Reviews absinkt. In Kombination mit der *total* und *average fixation duration* lässt sich auch erkennen, dass die Dauer der Fixationen der Novizinnen und Novizen im Verlauf des Reviews kürzer wird. Im Gegensatz dazu sind sowohl die *total* und *average number of fixations*, als auch die *total* und *average fixation duration* für die Expertinnen und Experten über die Phasen hinweg eher gleich verteilt und es ergeben sich lediglich marginale Unterschiede. Eine entsprechende Interpretation dieser Metriken gestaltet sich als schwierig, da sie sich nur bedingt so verhalten, wie in der Arbeit von Sheridan und Reingold (2017) beschrieben. Auch die typischen Charakteristiken der *holistic models of image perception* (Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017) lassen sich so nicht direkt beobachten. Ebenso sind die Unterschiede zwischen Expertinnen und Experten, sowie Novizinnen und Novizen lediglich marginal ausgeprägt. Für die phasenbasierte Betrachtung der *total* und *average number of fixations*, sowie für die *total* und *average fixation duration* kann somit keine finale Aussage getätigt werden, welche einen direkten Bezug zu den *holistic models of image perception* herstellen würde.

6.5 Einschränkungen

Wie alle Studien, weist das hier dargestellte Experiment zu Code Reviews in der Programmiersprache C++ mehrere Schwächen auf. Diese sollen im Anschluss näher betrachtet und diskutiert werden.

6.5.1 Eye-Tracking-Hard- und Software

Der Umstieg von SMI auf Eye-Tracker von Tobii, welcher im Rahmen dieser Studie vollzogen wurde, ging mit Komplikationen einher. Trotz der insgesamt besseren Hardwareleistung im Vergleich zum früher verwendeten SMI 250RED mobile zeigte sich, dass der Spectrum bzw. die verwendete Experimentalsoftware Tobii Pro Lab werksseitig keine Saccadenlänge ausgibt und diese lediglich über einen Workaround ermittelt werden könnte (Kanojia, 2020; Nyström et al., 2021; Tobii Pro, 2021). Dieser

Workaround würde jedoch vollen Zugriff auf die ungefilterten Rohdaten des Eye-Trackers erfordern, welche in dieser Form nicht abgerufen werden konnten. Es zeigte sich, dass die verwendete Version von Tobii Pro Lab in diesem Fall immer eine Vorfilterung durchgeführt hatte (Kanojia, 2020). Insofern konnte die *saccade length* (alternativ auch häufig als *saccade amplitude* bezeichnet) nicht für die Datenanalyse verwendet werden. Ebenso zeigte sich bei der Sichtung der Daten, dass es bei einigen Probanden Lücken gab. Dabei handelte es sich nicht um kurze Unterbrechungen, die auftreten würde, wenn eine Person den Kopf vom Eye-Tracker wegbewegt oder störende Lichteinfälle auftreten. Die gefundenen Lücken deckten teilweise Zeitspannen von bis zu zwei Minuten auf. Weder im Verlauf des Experiments, noch im betrachteten Gazeplot konnten diese beobachtet werden. Folglich scheint es sich bei diesem Datenverlust um ein Problem in der Experimentalsoftware zu handeln. Ein ähnliches Problem konnte auch für die Tobii Pro Glasses 2 beobachtet werden (Tobii Pro, 2021). Im Falle der vorliegenden Dissertation führte die beobachteten Lücken dazu, dass die Datensätze von sechs Versuchspersonen ausgeschlossen werden mussten.

6.5.2 Stichprobengröße und -eigenschaften

Obwohl diese Studie im Vergleich zu den zuvor in Kapitel 3 dargelegten Forschungsstand eine verhältnismäßig große Stichprobe untersucht, stellt sich diese in vielerlei Hinsicht dennoch als zu klein dar. Dies zeigt sich daran, dass lediglich ein Teil der grundlegenden Eye-Tracking-Metriken statistisch signifikante Ergebnisse in der Analyse erzielen.
Ebenso zeigt sich die Zusammensetzung der Gruppe der Expertinnen und Experten als eher problematisch. Da durch die vorhandenen Netzwerke der OTH Regensburg zum Teil Programmiererinnen und Programmierer aus Standardisierungsgremien zur Sprache C++ rekrutiert werden konnten, erweist sich der Begriff der Experten im Falle dieser Studie als sehr weitläufig. Vielmehr hätte eine Zwischengruppe gebildet werden können, welche als *Intermediates* fungieren. Eine weitere Rekrutierung dieser Versuchspersonen scheiterte jedoch im Zuge der Corona-Pandemie. Weiterhin bedingte diese Problematik, dass eine dritte Gruppe die Vergleichbarkeit zu Studie 1 (siehe Kapitel 5 verkompliziert hätte und zur Folge gehabt hätte, dass die Stichprobe für diese Studie deutlich unausgeglichener gewesen wäre. Dennoch sollten künftige Arbeiten auch diese Gruppe stärker fokussieren.

6.5.3 Codebeispiele und deren Angemessenheit

Wie auch in der Vorgängerstudie, griff auch die C++-Studie auf relativ kurze und eher schlichte Codebeispiele zurück. Diese waren zwar länger und komplexer gehalten als die Quellcodes, welche in Kapitel 5 beschrieben wurden, muten aber im Vergleich zu real existierenden Softwareprojekten immer noch sehr künstlich und konstruiert an. Dies bedingt sich auf der einen Seite dadurch, dass seitens der Eye-Tracker keine Scrollkompensation verfügbar ist und der Code auf einer Seite darstellbar sein muss (Tobii Pro, 2020, 2021), auf der anderen Seite ergibt sich die Vorausset-

zung, dass die Quellcodes auch für Studierende geeignet sein müssen. Im Vergleich zur Arbeit von Begel und Vrzakova (2018), welche reale Codes genutzt haben, ergibt sich insofern die Frage, wie authentisch die Stimuli in dieser Studie sind?
Weiterhin stellt sich vor allem in Bezug auf die Experten die Frage, ob die relativ kurzen Beispiele überhaupt deren Fähigkeiten voll ausreizen und sie wirklich ihr gesamtes Fähigkeitenspektrum für die Durchführung des Reviews nutzen müssen? Im Experiment zeigte sich, dass diese zum Teil sogar Kapazitäten frei hatten und Auffälligkeiten in Bezug auf den (veralteten) Styleguide dieser Codes identifizierten, die aus Sicht des Experiments nicht relevant (aber vorhanden) waren.

6.6 Überprüfung der Hypothesen

Auf die Diskussion der Ergebnisse folgend und unter Berücksichtigung der genannten Einschränkungen wird in den nachfolgenden Absätzen überprüft, welche der in Kapitel 4 genannten Hypothesen zur C++-Studie (siehe Absatz 4.3 bestätigt oder widerlegt werden kann. Diesbezüglich wird erneut zweistufig vorgegangen. In einem ersten Schritt werden die allgemeinen Hypothesen zu den verschiedenen Eye-Tracking-Metriken besprochen (siehe Absatz 6.6.1), im Anschluss daran deren phasenbasiertes Pendant (siehe Absatz 6.6.2).

6.6.1 Überprüfung der allgemeinen Hypothesen

Die Überprüfung der allgemeinen Hypothesen förderte signifikante Unterschiede für vier betrachtete Metriken zu Tage. Unter Berücksichtigung der Diskussionsergebnisse (siehe Absatz 6.4.1), sowie der limitierenden Faktoren in Bezug auf diese Studie (siehe 6.5), fasst die nachfolgende Tabelle 6.38 die Resultate zu den untersuchten Metriken zusammen und stellt dar, welche Hypothesen verworfen oder beibehalten werden.

Tabelle 6.38: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Anzahl der gefundenen Fehler	H1	Die Anzahl der gefundenen Fehler während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten. Experten finden signifikant mehr Fehler.

Tabelle 6.38: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Total number of fixations	Ho	Die *total number of fixations* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.
Average number of fixations	Ho	Die *average number of fixations* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.
Total fixation duration	Ho	Die *total fixation duration* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.
Average fixation duration	Ho	Die *average fixation duration* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich nicht signifikant zwischen Novizen und Experten. Es zeigen sich lediglich marginale Unterschiede zwischen den Gruppen.
Fixation rate	H1	Die *fixation rate* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen den Novizen und Experten. Die Experten weisen einen signifikant höheren Wert für die *fixation rate* auf.

Tabelle 6.38: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Total number of saccades	H1	Die *total number of saccades* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten. Für die Experten zeigt sich ein signifikant niedrigerer Wert für die *total number of saccades*.
Average number of saccades	H1	Die *average number of saccades* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich signifikant zwischen Novizen und Experten. Für die Experten zeigt sich ein signifikant niedrigerer Wert für die *average number of saccades*.
Total number of visits on erroneous lines	H0	Die *total number of visits on erroneous lines* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen erhöhter Median.
Average number of visits on erroneous lines	H0	Die *average number of visits on erroneous lines* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen erhöhter Median.
Total dwell time on erroneous lines	H0	Die total *dwell time on erroneous lines* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen erhöhter Median.

Tabelle 6.38: Überprüfung der allgemeinen Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothese	Resultat
Average dwell time on erroneous lines	Ho	Die *average dwell time on erroneous lines* während eines Code Reviews in der Programmiersprache C++ unterscheidet sich zwischen Novizen und Experten nicht signifikant. Für die Experten zeig sich ein im Vergleich zu den Novizen erhöhter Median.

Nach Auflistung der zutreffenden Hypothesen und der vorangegangen Diskussion kann an dieser Stelle die bei Absatz 4.3.1 formulierte Forschungsfrage *"Wie unterscheiden sich Novizen von Experten hinsichtlich ihrer Augenbewegungen während eines Code Reviews in der Programmiersprache C++ im Allgemeinen?"* beantwortet werden. Die Antwort ist, dass sich die beiden Gruppen insgesamt nicht allzu stark voneinander unterscheiden. Es zeigen sich jedoch für die Experten Trends ab, die darauf hindeuten, dass diese über elaboriertere Strategien verfügen und ihr Vorwissen besser in das Review einfließen lassen. Dies resultiert in einer effizienteren Durchführung des Reviews in Bezug auf die korrespondierenden Eye-Tracking-Metriken und einer besseren Fehlererkennung.

6.6.2 Überprüfung der phasenbasierten Hypothesen

Die Überprüfung der phasenbasierten Hypothesen zeigt Unterschiede bei zehn der elf untersuchten Metriken auf. Die akzeptierten Hypothesen bzw. die zutreffenden Aussagen werden nachfolgend unter Berücksichtigung der zuvor genannten Diskussionsergebnisse (siehe Absatz 6.4), sowie den angesprochenen Limitierungen dieser Studie (siehe Absatz 6.5) in Tabelle 6.39 dargestellt.

Tabelle 6.39: Überprüfung der phasenbasierten Hypothesen zum Verhalten der Eye-Tracking-Metriken während der Durchführung eines Code Reviews in der Programmiersprache C++

Abhängige Variable	Zutreffende Hypothesen	Resultate
Total number of fixations	H1	Die total number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++. Unterschiede finden sich vorrangig bei der Gruppe der Novizen.

	H1	Bei der Gruppe der Novizen fällt die total number of fixations für die dritte Phase signifikant niedriger aus. Für Experten kann kein signifikanter Unterschied festgestellt werden.
Average number of fixations	H1	Die average number of fixations der einzelnen Phasen unterscheidet sich bei einem Review in der Programmiersprache C++. Unterschiede finden sich vorrangig bei der Gruppe der Novizen.
	H1	Bei der Gruppe der Novizen fällt die average number of fixations für die dritte Phase signifikant niedriger aus. Für Experten kann kein signifikanter Unterschied festgestellt werden.
Total fixation duration	H1	Für die total fixation duration können bei einem Review in der Programmiersprache C++ signifikante Unterschiede zwischen den Phasen festgestellt werden. Diese finden sich für Experten und Novizen.
	H1	Für beide Gruppen zeigen sich signifikante Unterschiede bezüglich der dritten Phase. Diese fällt in beiden Fällen kürzer aus. Die Novizen weisen übergreifend eine insgesamt kürzere total fixation duration auf.
Average fixation duration	H0	Es können hinsichtlich der average fixation duration keine phasenbasierten Unterschiede für Experten und Novizen beobachtet werden.
	H0	Es zeigen sich keine signifikanten Unterschiede zwischen den Experten und Novizen bezüglich der average fixation duration in den untersuchten Phasen. Beide Gruppen weisen ähnliche Werte auf.
Fixation rate	H1	Die phasenbasierte Betrachtung der fixation rate bei einem Review in der Programmiersprache C++ zeigt Unterschiede zwischen allen Phasen für Experten und Novizen auf.
	H1	Beide Gruppen zeigen in den betrachteten Phasen ein ähnliches Verhalten. Insgesamt fällt die fixation rate für die Gruppe der Experten in Phase 2 und 3 höher aus, als für die Novizen.

Total number of saccades	H1	Die phasenbasierte Betrachtung der total number of saccades bei einem Review in der Programmiersprache C++ zeigt Unterschiede auf. Für beide Gruppen zeigen sich in Phase 3 signifikant mehr Unterschiede als in den anderen Phasen.
	H1	Die total number of saccades fällt für Experten in der dritten Phase deutlich niedriger aus als für Novizen.
Average number of saccades	H1	Bezüglich der phasenbasierte Betrachtung der average number of saccades bei einem Review in der Programmiersprache C++ zeigen sich Unterschiede. Bei beiden Gruppen unterscheidet sich Phase 3 signifikant zu Phase 1 und 2.
	H1	Die average number of saccades fällt für Experten in der dritten Phase deutlich niedriger aus als für Novizen
Total number of visits on erroneous lines	H1	Für die Experten zeigt sich bei der phasenbasierten Betrachtung der total number of visits on erroneous lines ein signifikanter Unterschied zwischen der ersten und zweiten Phase. Es können keine Unterschiede für Novizen beobachtet werden.
	H1	Bei der Durchführung eines Reviews in der Programmiersprache C++ fällt die total number of visits on erroneous lines für Experten in der ersten Phase deutlich geringer aus als in den beiden anderen Phasen. Die Novizen zeigen während des Reviews durchgängig relativ konstante Werte.
Average number of visits on erroneous lines	H1	Für die Experten zeigt sich bei der phasenbasierten Betrachtung der average number of visits on erroneous lines ein signifikanter Unterschied zwischen der ersten und zweiten Phase. Es können keine Unterschiede für Novizen beobachtet werden.

	H1	Bei der Durchführung eines Reviews in der Programmiersprache C++ fällt die average number of visits on erroneous lines für Experten in der ersten Phase deutlich geringer aus als in den beiden anderen Phasen. Die Novizen zeigen während des Reviews durchgängig relativ konstante Werte.
Total dwell time on erroneous lines	H1	Hinsichtlich der phasenbasierten Betrachtung der total dwell time on erroneous lines ergeben sich bei einem Code Review in der Programmiersprache C++ signifikante Unterschiede für Experten. Es können keine Unterschiede für Novizen festgestellt werden.
	H1	Bei Experten fällt die total dwell time on erroneous lines in der ersten Phase geringer aus, als im Vergleich zu Novizen. Diese zeigen während des Reviews verhältnismäßig konstante Werte.
Average dwell time on erroneous lines	H1	Hinsichtlich der phasenbasierten Betrachtung der average dwell time on erroneous lines ergeben sich bei einem Code Review in der Programmiersprache C++ signifikante Unterschiede für Experten. Es können keine Unterschiede für Novizen festgestellt werden.
	H1	Bei Experten fällt die average dwell time on erroneous lines in der ersten Phase geringer aus, als im Vergleich zu Novizen. Diese zeigen während des Reviews verhältnismäßig konstante Werte.

6.7 Zwischenfazit zu Studie 2

Die in diesem Kapitel dargestellte Studie zeigt, dass es bei Reviews in der Programmiersprache C++ zu erfahrungsbedingten Unterschieden zwischen den betrachteten Gruppen kommt und das Review in Phasen abläuft. Obwohl die Unterschiede zwischen den Experten und den Novizen bei der allgemeinen Betrachtung der Eye-Tracking-Metriken nur selten statistische Signifikanz erreichen, zeichnet sich dennoch der Trend ab, dass erfahrenere Versuchspersonen die Aufgaben effizienter lösen. Diese Versuchsteilnehmerinnen und -teilnehmer müssen zur Durchführung der Reviews weniger visuellen Aufwand investieren und identifizieren dennoch mehr Fehler. Auch das Verhalten der Eye-Tracking-Metriken ist im Falle der allgemeinen

Betrachtung sehr ähnlich zu den Charakteristiken der *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980) und den Eigenschaften von visueller Expertise (Gegenfurtner et al., 2011; Kok, 2016). Die phasenbasierte Betrachtung der Eye-Tracking-Metriken liefert hingegen deutliche Hinweise auf das Vorhandensein der verschiedenen Elemente der *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980). Basierend auf den analysierten Daten zeigen sich die Charakteristiken einer globalen und fokalen Betrachtung der Codebeispiele für die Expertinnen und Experten. In Kombination mit der besseren Fehlererkennung, sowie den AOI-Metriken und deren Veränderungen im Verlauf des Experiments zeigt sich weiterhin auch, dass das Vorwissen der Versuchspersonen über den Aufbau und die Struktur eines C++-Quellcodes einen Einfluss auf deren Betrachtungsstrategie ausübt (Kundel et al., 2007; Swensson, 1980).

Dennoch stellen sich die erlangten Ergebnisse nicht so klar dar, wie ursprünglich erhofft. Eine mögliche Ursache dafür könnte sich in der sich wiederholenden Anwendung der Eye-Movement-Pattern liegen. Diese wird im *global-focal search model* (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010) bereits postuliert und in der Arbeit von Begel und Vrzakova (2018) ebenfalls für Codebeispiele beobachtet. Eine Folge für die aufgabenübergreifende und eher summativ gehaltene Analyse ist daher, dass sich die relevanten Eye-Tracking-Metriken ausmitteln und signifikante Auffälligkeiten eher in der Datenmenge verwischen.

Insgesamt kann festgehalten werden, dass die *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017; Swensson, 1980) eine durchaus geeignete Interpreationsgrundlage für Augenbewegungen bei einem Code Review darstellen. Vor allem die aus Sicht des *global-focal search model* geschilderte globale Betrachtung eines Stimulus findet sich bei Quellcodes in Form eines Scans (Sharif et al., 2012; Uwano et al., 2006) wieder. Für die fokale Betrachtung müssten im Rahmen zukünftiger Studien jedoch weitere Metriken (z.B. die Länge von Saccaden, sowie deren Richtung) hinzugezogen werden, welche eher aus der Leseforschung stammen (Busjahn et al., 2015; Strukelj & Niehorster, 2018).

Kapitel 7

Diskussion der Ergebnisse

Die im Rahmen von Studie 1 (siehe Kapitel 5) und Studie 2 (siehe Kapitel 6) erlangten Ergebnisse werden nachfolgend in diesem Kapitel diskutiert und mit den Erkenntnissen von anderen Forscherinnen und Forschern verglichen. Dabei werden die Ergebnisse aus verschiedenen Perspektiven betrachtet, erklärt und entsprechend eingeordnet (siehe Absatz 7.1). Weiterhin wird auch auf die Grenzen und Probleme der im Kontext dieser Dissertation durchgeführten Studien eingegangen (siehe Absatz 7.2). Darauf folgend werden die in Kapitel 4 formulierten Forschungsfragen aufgegriffen und unter Berücksichtigung der erlangten Erkenntnisse und vorhandenen Grenzen beantwortet (siehe Absatz 7.3. Abschließend wird in Absatz 7.4 darauf eingegangen, wie die Ergebnisse dieser Dissertation zur Modellbildung beitragen können.

7.1 Betrachtung und Einordnung der erlangten Erkenntnisse

Sowohl in Studie 1 (siehe Kapitel 5) und Studie 2 (siehe Kapitel 6) machten sich unabhängig von der jeweils verwendeten Programmiersprache) Unterschiede zwischen den Experten und Novizen bemerkbar. Selbst wenn diese nicht durchgängig eine statistische Signifikanz erreicht haben, zeigte sich dennoch der Trend, dass die erfahreneren Versuchspersonen die Aufgaben effizienter und mit mehr Erfolg bearbeitet haben. Weiterhin zeigten sich in beiden Studien verschiedene visuelle Strategien bezüglich der Durchführung der Code Reviews.
Nachfolgend sollen diese Unterschiede aus verschiedenen Perspektiven betrachtet werden. Den Anfang macht diesbezüglich die Expertiseforschung (siehe Absatz 7.1.1). Dieser folgend wird der Aspekt der visuellen Expertise aufgegriffen, indem die aus den beiden Studien erlangten Ergebnisse auf Basis der *holistic models of image perception* betrachtet werden (siehe Absatz 7.1.2). Ergänzend zu diesen wird noch auf die *Cognitive Load Theory* zurückgegriffen (siehe Absatz 7.1.4)

7.1.1 Betrachtung aus Sicht der Expertiseforschung

Betrachtet man die Ergebnisse von Studie 1 (siehe Kapitel 5) und Studie 2 (siehe Kapitel 6) übergreifend aus Sicht der Expertiseforschung, so lassen sich mehrere Beob-

achtungen machen, die die Annahmen dieser Forschungsdisziplin bestätigen. Dabei handelt es sich konkret um die Expertenleistung und die Problemlösefähigkeit. Diese werden nachfolgend genauer erläutert.

Expertenleistung

Übergreifend über beide Studien zeigt sich, dass die Experten in den Code Reviews signifikant besser abschneiden, als die Novizen. In beiden Studien konnten sie im Vergleich eine deutlich höhere *Anzahl an Fehlern* identifizieren. Dies mag auf den ersten Blick nicht allzu verwunderlich wirken, da die Expertenleistung eines der zentralen Merkmale von Expertise (Ericsson, 2006a, 2006b, 2016; Ericsson et al., 1993; Ernst et al., 2016) bzw. des Expertenbegriffs darstellt (Gruber, 2007; Gruber, Jansen, Marienhagen & Altenmüller, 2010). Dennoch bedarf die Expertenleistung einer Erklärung im Kontext des Software Engineering und der Code Reviews.
Für die Durchführung eines erfolgreichen Code Reviews spielt die Anwendung von domänenspezifischen Wissen eine zentrale Rolle (Bacchelli & Bird, 2013; Edmundson et al., 2013). Dieses wird von den Experten im Laufe ihrer Ausbildung, durch eigenständige Beschäftigung mit entsprechenden Themen und auch im Rahmen ihrer beruflichen Tätigkeit erworben (Abran et al., 2014; Billett et al., 2018; Hauser, Stark et al., 2020) und entsprechend eingesetzt. Betrachtet man beispielsweise die bestehenden Korrelationskoeffizienten zwischen der *Anzahl der gefundenen Fehler* und der *allgemeinen*, sowie der *professionellen Programmiererfahrung* in den beiden durchgeführten Studien, so zeigen sich signifikante Zusammenhänge zwischen diesen Variablen. Umso erfahrener die Reviewer sind, umso mehr Fehler entdecken sie.
Ebenfalls deuten verschiedene Eye-Tracking-Metriken in der phasenbasierten Analyse (z.B. *fixation rate, number of visits on erroneous lines, total fixation duration, ...*, sowie die deskriptive Statistik zu diesen übergreifend über beide Studien darauf hin, dass die untersuchten Experten über elaboriertere Strategien verfügen, welche ihnen eine im Vergleich zu den Novizen effizientere Durchführung der Code Reviews ermöglichen (Ericsson, 2016; Ericsson & Towne, 2010; Simon & Chase, 1973). Ergänzt werden diese Strategien durch eine gesteigerte Abstraktionsfähigkeit, welche sich darin äußert, dass Experten mentale Modelle des Codes erstellen und diese zur vereinfachten Navigation nutzen können (Abid et al., 2019), was ebenfalls an den Eye-Tracking-Metriken der phasenbasierten Analyse ablesbar ist. Weiterhin zeigt sich, dass die Verwendung von mentalen Modellen nicht auf Code Reviews beschränkt ist, sondern im Software Engineering in mehreren Teilbereichen zu finden ist (Hauser et al., 2019; Hutzler et al., 2018). Insgesamt können sowohl die Strategien (beispielsweise bei der C++-Studie mit der jeweiligen *main method* des Quellcodes zu beginnen um einen ersten Überblick zu erlangen), als auch die gesteigerte Abstraktionsfähigkeit, sowie der Nutzen von mentalen Modellen als ein Resultat der jahrelangen Beschäftigung mit diesem Themenbereich betrachtet werden (Billett et al., 2018; Hauser, Stark et al., 2020).
Diese jahrelange Beschäftigung mit dem Themenbereich der Code Reviews in ver-

schiedenen Programmiersprachen führt auf Seiten der Experten zur Ausbildung von Heuristiken und Automatismen (Ericsson, 2016; Ericsson & Towne, 2010; Hauser et al., 2019; Hutzler et al., 2018; Simon & Chase, 1973). Werden Experten beispielsweise mit bestimmten Problemen oder Auffälligkeiten bei einem Codebeispiel konfrontiert, so werden diese automatisch ausgelöst und es kann eine entsprechende Lösung generiert bzw. angewandt werden. Diese Heuristiken und Automatismen spielen hinsichtlich der Lösung von domänenspezifischen Problemen eine entscheidende Rolle. Im nachfolgenden Absatz (7.1.1) soll speziell darauf eingegangen werden, wie Experten bei Code Reviews als Problemlöser fungieren.

Experten als Problemlöser

Zwar nicht allzu überraschend, aber durch die hier durchgeführten Studien bestätigt, zeigt sich, dass Experten auch bei Code Reviews über gute Fähigkeiten zur Problemlösung verfügen. Code Reviews stellen an und für sich bereits Aufgaben dar, welche stark auf das Thema der Problemlösung fokussiert sind (Baum et al., 2016, 2017; Beller et al., 2014; Boehm & Basili, 2001; Sommerville, 2012; Tufano et al., 2021), insofern verwundert es nicht, dass Experten diesbezüglich ihr domänenspezifisches Wissen einsetzen können und durch dieses einen entscheidenden Vorteil erlangen.
Bemerkbar machen sich die Problemlösefähigkeiten in erster Linie bei der *Anzahl der gefundenen Fehler*. Diese fällt übergreifend über beide Studien immer signifikant besser für die Experten aus. Sie sind sowohl in der C-, als auch in der C++-Studie dazu in der Lage, mehr Fehler als Novizen zu identifizieren. Auch die bestehenden Korrelationskoeffizienten zwischen der *Anzahl der gefundenen Fehler* und der *allgemeinen*, sowie *professionellen Programmiererfahrung* belegen einen Zusammenhang zwischen der Problemlösefähigkeit und der angesammelten domänenspezifischen Erfahrung hin.
Wie sich die Erfahrung exakt auswirkt und die Durchführung eines Reviews beeinflusst, kann nicht mit absoluter Gewissheit gesagt werden und müsste durch entsprechende Studien weiter untersucht werden, jedoch scheinen die in Kapitel 2, bei Absatz 2.2.3 dargelegten Charakteristiken von Experten auch auf die Domäne des Software Engineerings zuzutreffen. Durch das jahrelange Anhäufen von Erfahrungen im Bereich von Code Reviews, sowie die konstante Konfrontation mit Problemen und Fehlern, die beim Programmieren auftreten können, bauen die Experten eine gut strukturierte mentale Bibliothek von Fallbeispielen und Handlungsroutinen auf (Gruber, 2007). Dieses Wissen über Fehler und vulnerable Stellen im Quellcode wirkt sich darauf aus, wie das Review durchgeführt wird und macht Experten letztendlich zu guten Problemlösern (Bacchelli & Bird, 2013; Badampudi et al., 2019; Edmundson et al., 2013).

7.1.2 Betrachtung auf Basis der *holistic models of image perception*

Wie bereits in den vorangegangenen Absätzen dargelegt wurde, wirkt sich Wissen bzw. Expertise auf die Durchführung und den Ablauf eines Code Reviews aus. Dies

macht sich nicht nur an der bereits angesprochenen besseren Performance der Experten bemerkbar, sondern spiegelt sich auch in den Augenbewegungen wieder.
Diesbezüglich stellen die in Kapitel 2 angesprochenen *holistic models of image perception* (siehe das Unterkapitel zur visuellen Expertise bei2.3) nicht nur eine theoretische Grundlage dieser Arbeit dar, sie bilden auch eine Diskussionsgrundlage auf welcher die Unterschiede zwischen Experten und Novizen bei den durchgeführten Studien genauer betrachtet werden können. Diese Betrachtung soll nachfolgend für alle drei beschriebenen Modelle erfolgen und basierend auf diesem erfolgt eine übergreifende Einschätzung für die Anwendung der *holistic models of image perception* für Code Reviews oder das Software Engineering im Allgemeinen.

Betrachtung auf Basis des *global-focal search model*

Wie bereits im Theorieteil, wird auch im Rahmen der Betrachtung erneut mit dem *global-focal search model* von Nodine und Kundel (1987) begonnen. In all seinen verschiedenen Versionen (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010; Sheridan & Reingold, 2017) liegt diesem Modell die Annahme zugrunde, dass sich die Betrachtung eines visuellen Stimulus in eine *globale* und eine *fokale* Phase unterteilen lässt. Dieses phasenbasierte Vorgehen wurde in dieser Dissertation berücksichtigt, indem die Datensätze entsprechend der Bearbeitungszeit aufgeteilt wurden.
Nodine und Kundel (1987) charakterisieren die globale Phase vorrangig durch ihr Ziel: In dieser Phase soll sich ein Überblick über den zu betrachtenden Stimulus verschafft werden. Hinsichtlich der relevanten Eye-Tracking-Daten wirkt sich dies vor allem auf die *total* und *average number of fixations*, die *total* und *average fixation duration*, die *fixation rate*, sowie auf die *total* und *average number of saccades* aus (Gegenfurtner et al., 2011; Kok, 2016; Nodine & Kundel, 1987; Sheridan & Reingold, 2017). Von den genannten Metriken zeigen sich vor allem bei der *fixation rate* Anzeichen für die Annahmen des *global-focal search model*. In beiden durchgeführten Studien finden sich bei der *fixation rate* signifikante Unterschiede. In der C++-Studie fallen diese durch die durchgeführten Friedman-Tests auf, im Falle der C-Studie sind diese zwar vorhanden, aber weniger stark ausgeprägt. Interessant ist, dass für die *fixation rate* in beiden Studie ein identisches Muster ergibt: Sowohl für die Experten, als auch für die Novizen fällt diese in der ersten Phase immer am höchsten aus und nimmt im Verlauf des Reviews immer weiter ab. In der dritten Phase erreicht sie in beiden Studien ihren tiefsten Wert. Basierend auf der Metastudie von Sheridan und Reingold (2017) deutet das beobachtete Muster darauf hin, dass das Vorgehen der Versuchspersonen dadurch geprägt ist, dass sie sich anfangs einen Überblick über den zu begutachtenden Quellcode verschaffen wollen.
Entgegen der Annahmen zu den holistischen Modellen (Sheridan & Reingold, 2017) bzw. dem *global-focal search model* (Nodine & Kundel, 1987) lässt sich bezüglich der *total* und *average fixation duration* in beiden Studien kein signifikanter Unterschied zwischen den untersuchten Phasen feststellen. Es fällt lediglich auf, dass die Novizen bei der C-Studie deutlich niedrigere Werte aufweisen als die Experten. Die relativ

ähnliche Verteilung dieser Metrik über die Phasen hinweg ist insofern verwunderlich, da vor allem eine kurze *average fixation duration* in der ersten Phase mit einem Scan des zu begutachtenden Stimulus assoziiert wird (Nodine & Kundel, 1987; Sheridan & Reingold, 2017; Uwano et al., 2006).
Analog verhält es sich mit der *total* und *average fixation duration*. Diese zeigt ein ähnliches Muster, wie die Betrachtung der *fixation duration*.
Wendet man sich von der globalen Phase ab und konzentriert sich auf die fokale Phase des *global-focal search models* (Nodine & Kundel, 1987), so ergeben sich weitere Aspekte, die in in die Betrachtung einbezogen werden müssen. Nodine und Kundel (1987) charakterisieren diese vorrangig dadurch, dass in diesem Abschnitt zuvor identifizierte Auffälligkeiten näher untersucht werden. Aus einer datengestützten Perspektive zu den verschiedenen *holistic models of image perception* (Sheridan & Reingold, 2017) kommen diesbezüglich zu den bereits angesprochenen Metriken noch AOI-basierte Daten hinzu. In die phasenbasierte Auswertung der beiden durchgeführten Studien wurden daher die *total* und *average number of visits on erroneous lines*, sowie die *total* und *average dwell time on erroneous lines* aufgenommen. Diese Metriken beschäftigen sich damit, wie viel Zeit bzw. Fixiations für die Betrachtung von fehlerbehafteten Zeilen aufgewandt wird.
Bezüglich der *total* und *average number of visits on erroneous lines* ergeben sich sowohl zwischen den Phasen, als auch zwischen den Erfahrungsstufen signifikante Unterschiede. In der C-Studie unterscheidet sich die erste Phase signifikant von der zweiten und dritten Phase. Es kann beobachtet werden, dass die Werte für beide Metriken im Verlauf des Reviews zunehmen. Ebenso fallen die Werte für die Experten übergreifend über alle Phasen niedriger aus, als für die Novizen. In der C++-Studie lassen sich vorwiegend auf Seiten der Experten signifikante Unterschiede erkennen. Für sie unterscheidet sich die erste Phase signifikant von der zweiten. Ebenso zeigt sich der Trend aus der C-Studie, dass die *total* und *average number of visits on erroneous lines* in der ersten Phase am niedrigsten ausfällt. Für die Novizen in dieser Studie zeigt sich übergreifend über die Phasen eine relativ konstante Verteilung dieser Metrik. Es können keine relevanten Auffälligkeiten beobachtet werden.
Ein etwas abgeändertes Bild zeigt sich für die *total* und *average dwell time on erroneous lines*. In der C-Studie zeigen sich lediglich marginale Unterschiede zwischen den Phasen und diese finden sich auf Seiten der Novizen. Für die Experten zeigt sich, dass die *dwell time* in allen Phasen kürzer ausfällt als für die Novizen. Bei der C++-Studie kann für die Experten beobachtet werden, dass sie sich analog zur zuvor beschriebenen *total* und *average number of visits on erroneous lines* verhält. Diese weisen in der zweiten und dritten Phase deutlich erhöhte Werte auf. Für die untersuchten Novizen zeigen sich bei dieser Studie ebenfalls über alle Phasen hinweg relativ konstante Werte.
Nodine und Kundel (1987) merken ebenfalls an, dass das *global-focal search model* auch rekursive und serielle Prozesse beinhaltet. Während die seriellen Prozesse das Vorgehen in zwei Phasen zum Ausdruck bringen, zeigen sich die rekursiven Prozesse darin,

dass sich die globale und fokale Betrachtung abwechseln und sich während der Begutachtung eines Stimulus mehrfach wiederholen können, falls in diesem noch weitere Fehler vermutet werden (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000, 2010). Basierend auf den erlangten Daten macht sich dieser Prozess beispielsweise dadurch bemerkbar, dass die Versuchspersonen in beiden Studien immer wieder zu fehlerhaften Zeilen zurückkehren. Ebenso konnte in der C++-Studie beobachtet werden, dass Experten bei den beiden Beispielen mit drei Fehlern mehr Punkte erzielten als Novizen.

Führt man die Diskussion um das *global-focal search model* (Nodine & Kundel, 1987) und die erhobenen Daten fort und ergänzt diese mit den Erkennntnissen von anderen Forschern aus dem Software Engineering, so zeigt sich, dass auch diese ähnliche Beobachtungen gemacht haben (Begel & Vrzakova, 2018; Busjahn et al., 2011; Busjahn, Bednarik & Schulte, 2014; Busjahn et al., 2015; Nivala et al., 2016; Sharif et al., 2012; Uwano et al., 2006).

Die Arbeiten von Theresa Busjahn und ihren Kolleginnen und Kollegen (Busjahn et al., 2011; Busjahn, Bednarik & Schulte, 2014; Busjahn et al., 2015) beschäftigen sich intensiv mit der Durchführung von Code Reviews und den involvierten Augenbewegungen. In ihrer 2015 publizierten Arbeit *Eye movements in code reading: Relaxing the linear order* konnten sie belegen, dass sich das Leseverhalten bei diesen stark vom Lesen eines natürlichen Textes unterscheidet. Dieses ist bei Quellcode deutlich weniger linear und dadurch gezeichnet, dass sich die ausführenden Personen eher sprunghaft durch den zu begutachtenden Stimulus bewegen (Busjahn et al., 2011, 2015). Ähnliche Muster finden sich auch in der Leseforschung, werden aber nur in speziellen Fällen und meist nur durch spezielle Aufforderungen ausgelöst (Strukelj & Niehorster, 2018). Diese Sprünge nehmen mit zunehmender Erfahrung des Reviewers zu und das Leseverhalten entfernt sich mit ansteigendem Expertisegrad immer mehr von natürlichem Text. Funktional betrachtet orientieren sich die Sprünge am Ablauf des Codes und ermöglichen es so, dessen Inhalt und die dargestellte Struktur schneller zu erfassen (Busjahn et al., 2015).

Die von Busjahn et al. (2015) beschriebenen sprunghaften Bewegungen finden sich nicht nur in der hier vorliegenden Dissertation (Abid et al., 2019; Begel & Vrzakova, 2018; Jbara & Feitelson, 2015; Lin et al., 2016; Sharif et al., 2012; Uwano et al., 2006). Auch in der bereits erwähnten Arbeit von Nivala et al. (2016) konnten diese beobachtet und ausführlicher beschrieben werden. In dieser Studie zeigte sich klar, dass die Experten eine andere Strategie für die Durchführung des Reviews gewählt haben, als die Novizen. Verdeutlicht wird dies anhand der folgenden Abbildung 7.1. Diese stellt jeweils die ersten zehn Sekunden eines Code Reviews für einen Novizen (grün) und einen Experten (rot) gegenüber.

Abbildung 7.1 zeigt, dass sich die Vorgehensweisen im Sinne der von Busjahn et al. (2015) gemachten Beobachtungen unterscheiden. Die Novizen tendierten auch in dieser Studie (Nivala et al., 2016) dazu den Code ähnlich zu einem natürlichen Text zu lesen. Die Experten hingegen demonstrierten während der ersten zehn Sekunden

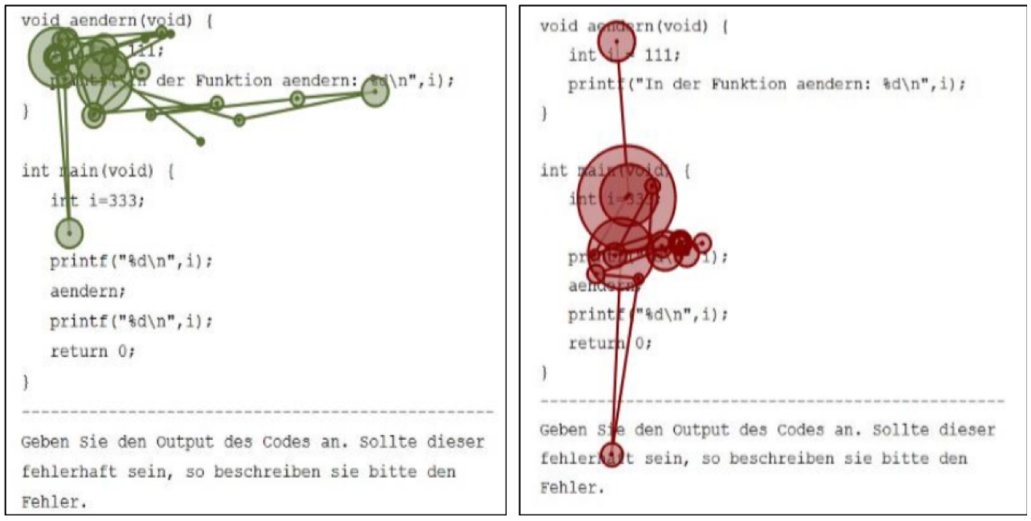

Abbildung 7.1: Gegenüberstellung der ersten 10 Sekunden der Durchführung eines Code Reviews für einen Novizen (rot) und Experten (grün) (übernommen von Nivala et al. (2016, S.618))

des Reviews einen initialen Scan des jeweiligen Codes. Dieser verfolgte das Ziel, sich über diesen einen Überblick zu verschaffen. Ebenso konnte beobachtet werden, dass irrelevante oder weniger kritische Areale des Codes ignoriert wurden oder weniger Aufmerksamkeit erhielten. Diese Ergebnisse (Nivala et al., 2016) und die Resultate der zuvor dargelegten Studien finden sich in dieser Form auch in den Arbeiten von anderen Forscherinnen und Forschern und legen die Bedeutung eines Scans, sowie die genaue Betrachtung von Auffälligkeiten für die erfolgreiche Durchführung eines Code Reviews nahe (Abid et al., 2019; Begel & Vrzakova, 2018; Jbara & Feitelson, 2015; Lin et al., 2016; Sharif et al., 2012; Uwano et al., 2006).

Betrachtung auf Basis des *two-stage detection model*

Wie bereits im Theorieteil dieser Dissertation (siehe Absatz 2.3 erwähnt wurde, weist das *two-stage detection model* von Swensson (1980) große inhaltliche Ähnlichkeiten zum *global-focal search model* (Nodine & Kundel, 1987; Nodine, Mello-Thoms, Kundel & Weinstein, 2002; Nodine & Mello-Thoms, 2010) auf. Insofern kann auf eine ausführliche Betrachtung der relevanten Eye-Tracking-Metriken verzichtet werden, da diese bereits durch die vorangegangenen Absätze abgedeckt worden ist. Dennoch unterscheidet sich der Ansatz von Swensson (1980) in zwei Punkten vom Modell von Nodine und Kundel (1987). Diese sollen noch thematisiert werden.
Swensson (1980) sieht in seinem Modell keine rekursiven Prozesse vor. Im *two-stage detection model* verläuft die Betrachtung eines visuellen Stimulus sehr linear. Auch wenn die Phasen große Ähnlichkeiten mit dem *global-focal search model* aufweisen geht Swensson nicht davon aus, dass sich diese im Verlauf der Betrachtung eines Stimulus wiederholen. Auf Basis der vorliegenden Daten und der Erkenntnisse anderer Autoren (Begel & Vrzakova, 2018; Busjahn et al., 2015; Nivala et al., 2016; Sharif et al.,

2012; Turner et al., 2014; Uwano et al., 2006) ist für Code Reviews jedoch davon auszugehen, dass sich der Wechsel zwischen globaler und fokaler Betrachtung mehrfach wiederholt und sogar ein entscheidender Bestandteil des Ablaufs ist. Insofern lässt sich diese Annahme des *two-stage detection model* im Kontext dieser Dissertation nicht aufrechterhalten.

Im Vergleich zum *global-focal search model* misst Swensson (1980) dem Vorwissen eine deutlich größere Bedeutung zu als Nodine und Kundel (1987). Sowohl die erhobenen Daten, als auch die Ergebnisse aus anderen Studien unterstützen diese Aussage: Auf der einen Seite lässt sich erkennen, dass beispielsweise das Leseverhalten der Versuchspersonen mit steigendem Expertisegrad sprunghafter wird und sich immer mehr an der Struktur des Quellcodes orientiert (Busjahn et al., 2015; Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Nivala et al., 2016; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006). Auf der anderen Seite kann auch beobachtet werden, dass weniger vulnerable Areale der Codebeispiele weniger Aufmerksamkeit erhalten oder ignoriert werden. Dies zeigt sich einmal in der bereits erwähnten Abbildung, welche aus der Arbeit von Nivala et al. (2016) übernommen wurde oder auch in der nachfolgenden Darstellung 7.2, welche den gesamten Gazeplot eines Experten und eines Novizen bei der Durchführung eines Reviews bei der hier besprochenen C++-Studie zeigt (Hauser, Schreistetter et al., 2020):

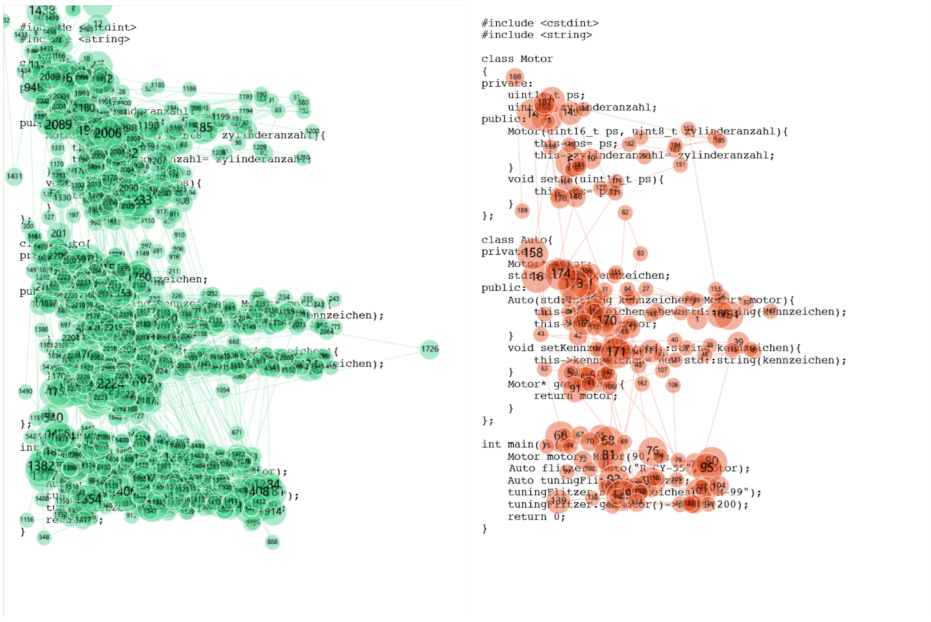

Abbildung 7.2: Gegenüberstellung der Vorgehensweisen eines Experten und eines Novizen bei der Durchführung eines Code Reviews in der Programmiersprache C++ (übernommen von Hauser et al. (2020b, S.3-4)

Zum anderen kann dieser Trend auch in Bezug auf die AOI-basierten Metriken erkannt werden. So

Betachtung auf Basis des *holistic mode vs. search to find*

Eine Betrachtung der Daten aus Sicht des *holistic mode vs. search to find* (Kundel et al., 2007) gestaltet sich in Bezug auf die Augenbewegungen bei Code Reviews eher als schwierig. Eine der zentralen Annahmen von Kundel et al. (2007) bezieht sich darauf, dass Experten in einer bestimmten Domäne mehr Informationen aus dem peripheren Sehen ziehen können. Durch diesen erweiterten Fokus wird der *holistic mode* erst zugänglich und ersetzt den langsameren Ansatz des *search to find*.
Auf Basis der erhobenen Daten kann keine Verifizierung des *holistic mode* erfolgen. Diese zielen nicht auf eine Untersuchung des peripheren Sehens ab. An dieser Stelle wären entsprechend geplante Studien notwendig, welche ihren Fokus auf die visuelle Informationsaufnahme von Experten bei Code Reviews legen. Diese müssten in ihrem Design auch darauf ausgelegt werden, das periphere Sehen gezielt zu analysieren um entsprechende Beobachtungen ableiten zu können (Duchowski, 2017; Holmqvist et al., 2011).
Dennoch enthält der Ansatz von Kundel et al. (2007) Erklärungspotenzial für diese Dissertation: Die Annahme, das Novizen bei der Betrachtung eines Quellcodes eher auf ein Vorgehen im Sinne von *search to find* zurückgreifen bestätigt sich auch in den durchgeführten Studien. In diesen ähnelt das Leseverhalten von Novizen mehr dem Lesen eines natürlichen Textes. Ähnliche Beobachtungen machten auch andere Forscher (Begel & Vrzakova, 2018; Busjahn et al., 2015; Nivala et al., 2016; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006). Diese Beobachtung kann auch den gesammelten Daten aus Studie 1 und 2 entnommen werden, in welchen sich durchgängig zeigte, dass Novizen weniger effizient performen als Experten.

7.1.3 Fazit zur Nutzung der *holistic models of image perception* zur Analyse von Augenbewegungen während eines Code Reviews

Die drei in dieser Dissertation vorgestellten *holistic models of image percetption* (Kundel et al., 2007; Nodine & Kundel, 1987; Swensson, 1980), sowie die dazu begleitenden Studien (Gegenfurtner et al., 2011; Gegenfurtner, 2020; Kok, 2016; Reingold & Sheridan, 2012; Sheridan & Reingold, 2017) stellen für für die Analyse und Interpretation von Augenbewegungen während eines Quellcodes insgesamt eine gute Grundlage dar. Den meisten Nutzen liefert diesbezüglich das *global-focal search model* von Nodine und Kundel (1987), welches eine phasenbasierte Vorgehensweise nahelegt. Mit der Unterteilung in eine globale und eine fokale Phase liefert es auch für die in den Studien dieser Dissertation beobachteten Augenbewegungen und gesammelten Daten eine Interpretationsgrundlage (Nodine & Kundel, 1987; Nodine & Mello-Thoms, 2000; Nodine et al., 2002; Nodine & Mello-Thoms, 2010). Gleichzeitig decken sich auch die Erkenntnisse anderer Forscher mit diesem Modell und weisen Gemeinsamkeiten auf. Hier wäre vor allem das Scan-Pattern zu Beginn der Code Reviews zu nennen, welches sich so in mehreren Arbeiten wiederfindet. Selbiges gilt für die Annahmen bezüglich der fokalen Phase und dem Ablauf des Reviews als einen rekur-

siven Prozess (Abid et al., 2019; Begel & Vrzakova, 2018; Busjahn et al., 2015; Hauser, Reiß, Nivala, Mottok & Gruber, 2017; Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Nivala et al., 2016; Sharif et al., 2012; Uwano et al., 2006). Auf das *two-stage detection model* (Swensson, 1980) treffen aufgrund der Ähnlichkeiten zum *global-focal search model* (Nodine & Kundel, 1987) ähnliche Aussagen zu. Kritisch anzumerken ist das Fehlen, der rekursiven Prozesse. Diese können sowohl durch die eigenen Daten, als auch durch die Arbeiten anderer Forscher (Begel & Vrzakova, 2018; Uwano et al., 2006) für Code Reviews belegt werden. Was es jedoch beisteuert, ist die Rolle des Einflusses von Vorwissen (Swensson, 1980). Die Filterwirkung welches dieses auf die visuelle Wahrnehmung bzw. die Vorgehensweise bei einem Review ausübt, wird weder im *global-focal search model* (Nodine & Kundel, 1987), noch beim Ansatz des *holistic modes vs. search to find* (Kundel et al., 2007) ausreichend gewürdigt. Gleichzeitig ergibt sich so eine Schnittstelle zur Expertiseforschung. Die Annahmen des *holistic mode vs. search to find* (Kundel et al., 2007) können zwar auf Basis der vorliegenden Daten nicht vollständig verifiziert werden, jedoch scheint die Annahme, dass Experten mehr Informationen aus dem peripheren Sehen erlangen können nicht abwegig. Hier bedarf es spezieller Studien, um ein eindeutiges Ergebnis zu erzielen. Das Vorgehen im Sinne von *search to find* (Kundel et al., 2007) kann jedoch belegt werden (Busjahn et al., 2015; Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Nivala et al., 2016). Die grundlegende Annahme, dass mit steigendem Expertisegrad auch die visuelle Wahrnehmung an die Anforderungen der jeweiligen Domäne angepasst werden findet sich bereits in den Definitionen für visuelle Expertise wieder (Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017), auf denen diese Dissertation basiert.

Betrachtet man die *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Swensson, 1980; Sheridan & Reingold, 2017) übergreifend, so haben alle drei verwendeten Modelle gemeinsam, dass sie durch ein phasenbasiertes Vorgehen mit jeweils charakteristischen Augenbewegungen gekennzeichnet sind. Je nachdem, welches Model gewählt wird, geht man bei der Analyse davon aus, dass es sich bei den Phasen um rekursive Prozesse (beispielsweise beim *global-focal search model* (Nodine & Kundel, 1987) oder dem *holistic mode vs. search to find* (Kundel et al., 2007)) oder um eine lineare Abfolge (im Falle von Swenssons (1980) *two-stage detection model*) handelt. Da Swenssons (1980) für Quellcode als nicht zutreffend erachtet wird, liegt die Annahme nahe, dass zwischen einer globalen und fokalen Betrachtungsweise während des Reviews mehrfach gewechselt wird (Aschwanden & Crosby, 2006; Begel & Vrzakova, 2018; Busjahn et al., 2015; Nivala et al., 2016; Turner et al., 2014; Uwano et al., 2006). Dieser Wechsel zwischen der Betrachtungsweise geht mit veränderten Augenbewegungen einher. Für eine gruppenbasierte Betrachtung, wie sie sich im Falle dieser Dissertation findet, ergibt sich daher als Konsequenz, dass bei jedem Wechsel ein *Rauschen* in den Metriken bzw. der statistischen Auswertung entsteht. Diese macht sich beispielsweise dadurch deutlich, dass verhältnismäßig wenige Er-

gebnisse statistische Signifikanz erreichen und hinsichtlich der deskriptiven Statistik zu den untersuchten Metriken sowohl in der allgemeinen, als auch in der phasenbasierten Betrachtung eine relativ große Spannweite abgedeckt wird.
Fasst man die Ergebnisse zu den *holistic models of image perception* zusammen, lassen sich auch die in Kapitel 4 formulierten Forschungsfragen beantworten.
Die erste Forschungsfrage diesbezüglich bei Absatz 4.4 lautete *Welches der holistic models of image perception eignet sich am besten für die Analyse und Interpretation von Augenbewegungen während eines Code Reviews?* Auf Basis der erlangten Erkenntnisse empfiehlt sich das *global-focal search model* von Nodine und Kundel (1987), da es die meisten Erkenntnisse für eine Interpretation der Augenbewegungen beisteuert. Weiterhin bietet es einen hohen Allgemeingülitgkeitsgrad, der eine Nutzung auch abseits von Bildmaterial ermöglicht.
Die zweite Forschungsfrage adressierte speziell die Komponenten der verschiedenen *holistic models of image perception* und lautet *Welche Komponenten der holistic models of image perception finden sich bei Code Reviews wieder?* Diesbezüglich können aus dem *global-focal search model* (Nodine & Kundel, 1987) die globale und fokale Betrachtung des Quellcodes, sowie die rekursiven Prozesse belegt werden. Ebenso zeigt sich dass das von Swensson (1980) thematisierte Vorwissen einen Einfluss auf die Durchführung der Reviews ausübt.
Abschließend wird bei der dritten Forschungsfrage nach einem vereinheitlichten Model zur Interpretation von Augenbewegungen während eines Code Reviews gefragt. Diese Frage soll nachfolgenden bei Absatz 7.4) genauer beantwortet werden.
Obwohl die *holistic models of image perception* (Kundel et al., 2007; Nodine & Kundel, 1987; Swensson, 1980) ursprünglich für die Analyse und Interpretation von Bildern oder grafischen Darstellungen entwickelt worden sind (Reingold & Sheridan, 2012; Sheridan & Reingold, 2017) und auch vorrangig in entsprechenden Domänen (beispielsweise der Radiologie) eingesetzt werden (Gegenfurtner et al., 2011, 2017; Gegenfurtner & Merriënboer, 2017; Kok, 2016; Kok, de Bruin, Robben & van Merriënboer, 2012), lässt sich als Gesamtfazit zu festhalten, dass die Modelle nicht auf diese Anwendung limitiert sind. Aufgrund der ihres Designs stellen sie sich als relativ flexibel dar und können auch in anderen, stark visuell geprägten Domänen genutzt werden. Obwohl sie für die Anwendung bei Code Reviews einiger Modifikationen bedürfen, stellen sie eine gute Grundlage dar um sich mit visueller Expertise in diesem Bereich zu beschäftigen. Insofern wird auch den Aufforderungen von Sharafi et al. (2015) und Obaidellah et al. (2018) Genüge getan, welche sich wünschen, dass elaborierte Theorien und Ansätze aus anderen Domänen im Software Engineering getestet, angepasst und genutzt werden.

7.1.4 Betrachtung auf Basis der *cognitive load theory*

Ergänzend zu den zuvor dargelegten Betrachtungsperspektiven der Expertiseforschung (siehe Absatz 7.1.1) und der *holistic models of image perception* (siehe Absatz 7.1.2) bietet die *cognitive load theory* (Sweller, 1988, 1994; Sweller et al., 1998) weitere

Ansätze, welche in die Erklärung der Expertenleistung einbezogen werden sollten. Folgt man der in Kapitel 2 beschriebenen *cognitive load theory* (Sweller, 1988, 1994; Sweller et al., 1998) und versucht die Leistung der Versuchspersonen auf die drei genannten *loads* abzubilden, würden sich für die Novizen und Experten zwei unterschiedliche Darstellungen ergeben. Basierend auf den erhobenen Daten werden diese und die dahinterstehenden Interpretationen nachfolgend in den Abbildung 7.3 und 7.4 dargestellt.

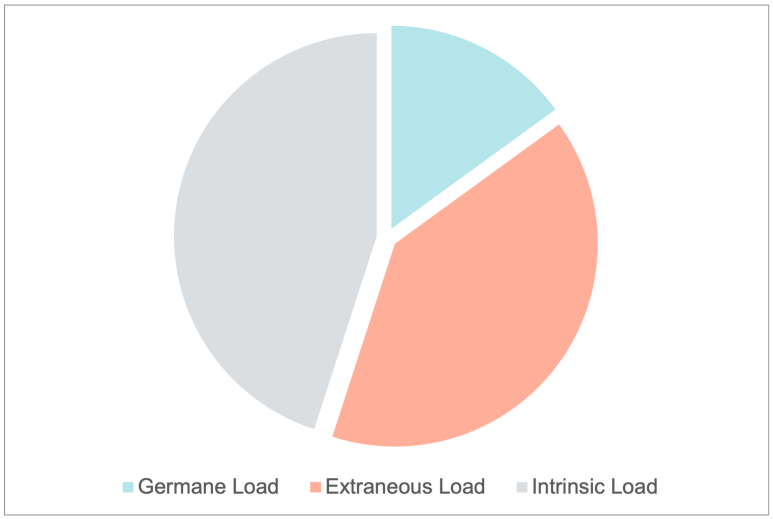

Abbildung 7.3: Schematische Aufteilung der verschiedenen *loads* der *cognitive load theory* für einen Novizen

Auf Seiten der Novizen ließe sich ein verhältnismäßig schwach ausgeprägter *germane load* beobachten. Sowohl der *intrinsic* und der *extraneous load* werden in Abbildung 7.3 als gleich oder zumindest ähnlich angesehen. Dies lässt sich damit begründen, dass ein Code Review von den ausführenden Personen bestimmte Fähigkeiten voraussetzt, welche im Rahmen des Expertiseerwerbs aufgebaut und verfeinert werden, sodass langfristig Automatismen und Heuristiken entstehen (Hauser et al., 2019; Hauser, Stark et al., 2020; Hutzler et al., 2018). Gleichzeitig kann aufgrund der Selbsteinschätzungen in den beiden durchgeführten Studien, sowie der Performance der Novizen davon ausgegangen werden, dass noch keine ausreichende domänenspezifische Erfahrung vorliegt um den *germane load* zu optimieren. Selbst Hilfestellungen in Form von Coding Guidelines, veränderte Darstellungsformen oder ein abgeändertes Syntaxhighlighting könnten sich auf diesem Erfahrungsniveau eher als hinderlich erweisen (Grabinger, Hauser & Mottok, 2023a, 2023b; Homann, Grabinger, Hauser & Mottok, 2023; Moser, 2022; Schorr, o. J.). Ebenso sollte bedacht werden, dass unter Umständen mit Ablenkungen, wie sie im Alltag eines Softwareentwicklers auftauchen können, noch wenig Erfahrung herrscht und sich diese nachteilhaft auf die Minimierung des *intrinsic load* auswirken können (Bacchelli & Bird, 2013; Edmundson et al., 2013; Kononenko et al., 2016).
Bezüglich der Experten kann von einem anderen Bild ausgegangen werden. Eine angenommene Verteilung der *loads* findet sich in Abbildung 7.4. Aufgrund der bereits im Expertiseteil angesprochenen domänenspezifischen Erfahrungen verfügt der

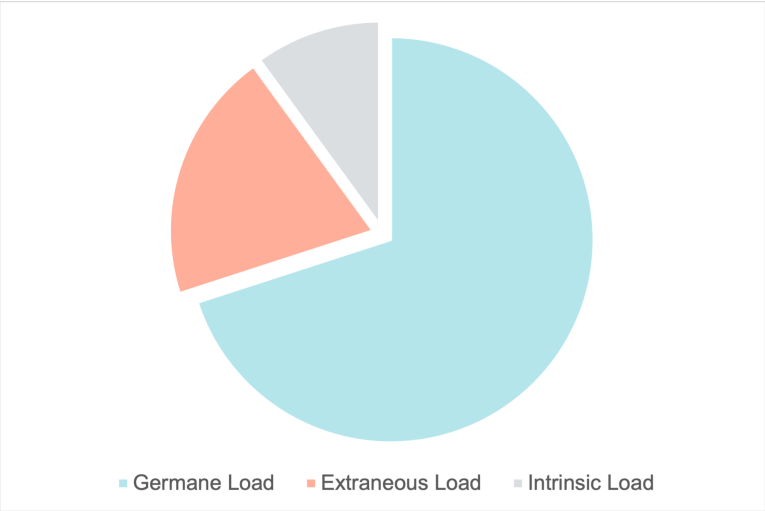

Abbildung 7.4: Schematische Aufteilung der verschiedenen *loads* der *cognitive load theory* für einen Experten

Experte bei der Durchführung eines Code Reviews bereits über Heuristiken und Automatismen, sowie ein fundiertes Wissen über die Struktur und Funktionsweise eines Quellcodes, sowie über Fehler und vulnerable Stellen (Begel & Vrzakova, 2018; Billett et al., 2018; Ericsson & Towne, 2010; Hauser, Stark et al., 2020). Dies minimiert zwar nicht den *intrinsic load*, gibt aber entsprechende Vorteile hinsichtlich der Durchführung eines Code Reviews. So kann beispielsweise besser zwischen relevanten und irrelavanten Details unterschieden werden und die zur Verfügung stehende Energie gezielt auf die Durchführung des Reviews bzw. das Finden von Fehlern gelenkt werden (Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006). Ebenso zeigte sich beispielsweise auch im Rahmen einer Studie, in deren Rahmen *Eye Movement Modeling Examples* erstellt worden sind, dass Experten bei der Erledigung der Aufgaben problemlos dem Ansatz des *think aloud* folgen können und ihr Vorgehen detailliert beschreiben können (Hauser, Grabinger, Mottok, Jahn & Nadimpalli, 2023). Bezüglich des *extraneous load* kann ebenfalls davon ausgegangen werden, dass dieser aufgrund der angesammelten domänenspezifischen Erfahrung reduziert werden kann (Bacchelli & Bird, 2013; Edmundson et al., 2013; Kononenko et al., 2016). Zusammenfassend zeigt sich auf Basis der Betrachtung durch die *cognitive load theory*, dass sich der Expertiseerwerb auf die Durchführung der Code Reviews auswirkt. Die Annahme, dass Experten relevante von irrelevanten Details besser unterscheiden können und ihre Aufmerksamkeit auf die Durchführung einer bestimmten Aufgabe bündeln können (Sweller, 1988, 1994; Sweller et al., 1998) ist bezüglich der Begutachtung von Quellcode und für das Software Engineering im Allgemeinen von essentieller Bedeutung (M. M. Moore, 2002; Reuter, Beslmeisl & Mottok, 2017; Sommerville, 2012; Soska, Mottok & Wolff, 2016). Dieser Gedanke fügt sich in die bereits zuvor geschilderte Betrachtung aus Sicht der Expertiseforschung ein. Hier sei auf die *information reduction hypothesis* verwiesen, welcher ein ähnlicher Gedanke zugrunde liegt (Haider & Frensch, 1996, 1999).

Obwohl die *cognitive load theory* in den Studiendesigns der hier vorliegenden Dissertation stärker hätte berücksichtigt werden können, ergänzt sie die bereits dargelegten Erklärungen um weitere Aspekte. Aufgrund der weiten Verbreitung dieser Theorie und ihres domänenneutralen Charakters finden sich zu ihr auch entsprechende Tools und Konzepte, welche Datenerhebungen relativ einfach gestalten (Hart & Staveland, 1988). Gleichzeitig bietet die *cognitive load theory* aus Sicht des Instructional Design ein breites Anwendungsspektrum für den Bereich der Code Reviews (Sweller, 1988). Unter Zuhilfenahme des Eye-Trackings könnten beispielsweise gezielt Komplikationen identifiziert und didaktisch aufbereitet werden.

7.2 Grenzen der durchgeführten Studien

Dieser Abschnitt der Dissertation geht näher auf die Schwachstellen und Grenzen der durchgeführten Studien ein. Es soll damit begonnen werden, dass die Authentizität und Eignung der verwendeten Codebeispiele, sowie auf diese bezogene limitierende Faktoren besprochen werden (siehe Absatz 7.2.1). Ebenso werden nachfolgend die in beiden Studien relativ geringen Stichprobengrößen (siehe 7.2.2), sowie das bereits angesprochene Rauschverhalten (siehe 7.2.3) adressiert. Weiterhin wird der Begriff des Experten im Kontext dieser Dissertation hinterfragt (siehe Absatz 7.2.4). Den Abschluss bildet eine kritische Betrachtung der durchgeführten Datenauswertungen (siehe Absatz 7.2.5).

7.2.1 Authentizität und Eignung der verwendeten Codebeispiele

Mögliche Kritikpunkte in Bezug auf die vorliegende Dissertation lassen sich vor allem hinsichtlich der verwendeten Codebeispiele finden. Übergreifend über beide Studien ergeben sich dabei mehrere Probleme, die vorrangig aus den Anforderungen an die Codebeispiele, sowie den technischen Limitierungen der eingesetzten Hard- und Software resultieren. Dieses Zusammenspiel wirkt sich vor allem auf die Authentizität der verwendeten Stimuli negativ aus.
Bei der Erstellung der Codebeispiel galt es studienübergreifend zu beachten, dass diese von Novizen, sowie von Experten bearbeitet werden sollen. Das Fähigkeits- bzw. Erfahrungsniveau der beiden Gruppen wirkt sich in diesem Fall jedoch relativ limitierend aus. So können beispielsweise nur Inhalte verwendet werden, welche in dieser Form bereits im Studium vermittelt werden und folglich bekannt und lösbar sind. Dies hat auch zur Konsequenz, dass die verwendeten Stimuli künstlich generiert werden müssen und nur bestimmte Fehlerarten, Konzepte und Standards (z.B. moderne Coding Guidelines aus Wirtschaft und Industrie, bestimmte Highlighting-Optionen um zusätzliche Informationen verfügbar zu machen, ...) verwendet werden können (Homann et al., 2023; Moser, 2022; Schorr, o. J.). Weiterhin gilt es zu beachten, dass die Codebeispiele möglichst domänenneutral bzw. allgemein gehalten werden müssen, was sich ebenfalls limitierend auf deren Ausgestaltung auswirkt. Daraus ergibt sich wiederum, dass es sich bei den Beispielen um Codes handelt,

die in dieser Form in einem professionellen Kontext mit sehr hoher Wahrscheinlichkeit nicht auftauchen würden. Diese Problematik (vorrangig die Tatsache, dass es sich um künstliche Codes handelt, die einer Coding Guideline aus der Hochschullehre folgen) wurde auch von erfahrenen Versuchspersonen der zuvor vorgestellten C++-Studie angesprochen (Hauser, Schreistetter et al., 2020) und zeigt sich auch in anderen Studienarbeiten, die bei der Erstellung ihrer Stimuli ähnliche Kompromisse eingehen (Jbara & Feitelson, 2015; Nivala et al., 2016; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006).
Einen limitierenden Einfluss auf die Erstellung der Stimuli hatten auch die technischen Gegebenheiten der verwendeten Hard- und Software. Letztgenannte stellte sich über den gesamten Projektverlauf als das komplexere Problem dar.
Von Seiten der Hardware gilt es zu beachten, dass die Codebeispiele so generiert werden sollten, dass eine möglichst hohe Genauigkeit erzielt werden kann. Dies kann gewährleistet werden, indem eine entsprechend große Schriftgröße (im Fall der vorliegenden Studien war diese immer >12pt) gewählt wird und Messungenauigkeiten des Eye-Trackers bestmöglich ausgeglichen werden. So neigten beispielsweise die bei Hauser et al. (2020b) verwendeten Tobii Spectrum in der damaligen Konfiguration dazu, zum Rand hin etwas ungenauer zu werden. Dies bedingt in der Folge, dass die Stimuli an einem bestimmten Punkt (möglichst mittig) pärsentiert werden sollten (Ezer, Greiner, Grabinger, Hauser & Mottok, 2023; Nyström et al., 2021; Tobii Pro, 2020).
Ausgeprägter stellen sich die Limitierungen der verwendeten Software dar. Die Daten der vorliegenden Arbeit wurden mit der jeweiligen Experimentalsoftware von SMI (Hauser, Reuter et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023) und Tobii (Hauser, Schreistetter et al., 2020) erhoben. Beide Toolsuites erfüllen zwar ihren Zweck, erfordern aber einige Kompromisse um im Kontext der vorgestellten Studien genutzt werden zu können. So verfügten beide Programme zum Zeitpunkt der Experimente (und darüber hinaus (Moser, 2022)) über keine ausreichende Scrollkompensation, was dazu führt, dass lediglich sehr kurze Codebeispiele verwendet werden können. Je nach Setup liegt das Limit bei ca. 60-70 Zeilen Quellcode. Dies stellt insofern ein Problem dar, da professionelle Entwicklerinnen und Entwickler in ihrem Berufsalltag mit Codes arbeiten, die sich über tausende oder gar Millionen Zeilen erstrecken können (Broy, 2006; Cha et al., 2019; Sillito et al., 2006) und selbst Studierende im Laufe ihrer Programmierausbildung mit komplexeren und längeren Aufgaben konfrontiert werden.
Mögliche Lösungsansätze, wie beispielsweise die Aufzeichnung über die Bildschirmaufnahme (Nivala et al., 2016) oder das Verwenden des integrierten Browser der Eye-Tracking-Software (Moser, 2022) stellten keine praktikablen Vorgehensweisen dar. So zeigte sich für die SMI-Software BeeGaze, dass Screenrecords nicht standardmäßig mit einer AOI-basierten Analyse ausgewertet werden können und diese (wenn überhaupt) lediglich über mehrere Umwege und einem Qualitätsverlust auf Seiten der Eye-Tracking-Daten durchgeführt werden kann (Nivala et al., 2016; Hauser, Reuter

et al., 2018; Hauser, Grabinger, Mottok & Gruber, 2023). Ebenso stellte sich beim Arbeiten mit Tobii ProLab heraus, dass dieses mit Softwarefehlern zu kämpfen hat. Beispielsweise kam es bei mobilen Datenerhebungen zu Ausfällen auf dem Eye-Tracking-Laptop. Obwohl die Daten aufgezeichnet wurden, konnte während des Experiments keine Echtzeitdarstellung der Augenbewegungen betrachtet werden, wenn der externe Monitor über ein HDMI-Kabel angeschlossen war (Hauser, Schreistetter et al., 2020).
Gravierender wirken sich jedoch die analysebezogenen Einschränkungen seitens Tobii ProLab aus. Bei der Auswertungen der Rohdaten der C++-Studie (Hauser, Schreistetter et al., 2020) zeigte sich, dass ProLab werksseitig nicht dazu in der Lage ist bestimmte Eye-Tracking-Metriken auszugeben. So werden beispielsweise vorrangig Metriken, die sich auf saccades beziehen zwar global für den gesamten Datensatz berechnet, die dafür notwendigen Daten aber nicht in den Rohdaten hinterlegt. Diese Thematik sorgt bereits seit längerer Zeit für Unmut in der Nutzergemeinschaft von Tobii ProLab und konnte bis zur Veröffentlichung dieser Dissertation nicht behoben werden. In entsprechenden Foren (z.B. Researchgate) werden mögliche Workaround-Lösungen diskutiert und stellenweise auch geteilt, bei denen beispielsweise die fehlenden Daten über Matlab berechnet oder entsprechende R-Packages hinzugezogen werden (Kanojia, 2020).
Im Falle der C++-Studie zeigte sich jedoch, dass die Filteroptionen der R-Packages ungenügend waren. Diese basierten auf der zeitbasierten Erkennung bzw. Filterung von Augenbewegungen, wohingegen der Rest der ausgegebenen Rohdaten geschwindigkeitsbasierte Filter durchlief. Als Konsequenz würde sich eine mangelhafte Vergleichbarkeit der Daten ergeben.

7.2.2 Geringe Stichprobenzahl und Gruppenunterteilung

Beide Studien greifen auf verhältnismäßig geringe Stichprobengrößen zurück. Bei der C-Studie konnten die Daten von 23 Personen genutzt werden, im Falle der C++-Studie die von 34 Probanden. Aus Sicht des in Kapitel 3 dargelegten Standes der Forschung zur visueller Expertise bei Code Reviews bewegen sich diese Größen jedoch im domänenüblichen Durchschnitt, der bei N=27.375 Versuchspersonen liegt (grafisch illustriert in Abbildung 3.4) (Obaidellah et al., 2018; Sharafi et al., 2015). Vom Standpunkt der klassischen Statistik aus betrachtet sind die Stichproben beider Studien allerdings als klein zu bezeichnen (Bortz & Schuster, 2010; Field et al., 2012). Weiterhin wurden die teilnehmenden Personen gemäß ihres Erfahrungslevels unterteilt um in Kleingruppen den kontrastiven Vergleichen unterzogen werden zu können. Diesbezüglich muss beachtet werden, dass die entstandenen Kleingruppen nicht ausbalanciert sind und die Experten (stärker ausgeprägt in der C-Studie (Hauser, Mottok & Gruber, 2018; Hauser, Grabinger, Mottok & Gruber, 2023)) unterrepräsentiert sind. Als Folge fehlt bei vielen der angewandten statistischen Verfahren die Kraft einer größeren Stichprobe, welche (vor allem in Bezug auf die allgemeine Betrachtung der Eye-Tracking-Daten) klarere Resultate produzieren würde (Bortz &

Schuster, 2010; Cohen, 1988, 1992; Field et al., 2012).
Dieses Problem zeigt sich bezüglich der Erforschung von visueller Expertise mittels Eye-Tracking häufiger (Sheridan & Reingold, 2017) und findet sich im Kontext entsprechender Studien im Software Engineering verstärkt wieder (Obaidellah et al., 2018; Sharafi et al., 2015). Dies liegt vorrangig an der (bereits zuvor angesprochenen) erschwerten Rekrutierung von Experten in dieser Domäne. Als probater Lösungsansatz präsentiert sich diesbezüglich der Ansatz, die Datenerhebungen vor Ort bei den Versuchspersonen durchzuführen (Bittner, Ezer, Grabinger, Hauser & Mottok, 2023; Hauser, Schreistetter et al., 2020).

7.2.3 Statistische Signifikanz und *Rauschen*

Aus statistischer Sicht stellt es sich als problematisch dar, dass ein großer Teil der Ergebnisse in den beiden durchgeführten Studien keine Signifikanz erreichen konnte und die beobachteten Unterschiede häufig nur marginal ausfallen. Dies hat mehrere Gründe, die im Folgenden dargelegt werden sollen.
Auf der einen Seite macht sich diesbezüglich die im vorherigen Absatz 7.2.2 thematisierte Stichprobenzahl bemerkbar. Diese fällt für Verfahren der klassischen Statistik bzw. den in der Expertiseforschung häufig genutzten kontrastiven Vergleich relativ klein aus (Bortz & Schuster, 2010; Cohen, 1988, 1992; Ericsson et al., 1993; Field et al., 2012).
Auf der anderen Seite zeigt sich ebenfalls, dass die Daten bei genauerer Betrachtung einen etwas verrauschten Eindruck machen. Dies wurde bereits im Rahmen der Betrachtung auf Basis der *holistic models of image perception* (siehe Absatz 7.1.2) erläutert und ist auf die Wechsel zwischen den verschiedenen Betrachtungsmodi zurückzuführen. Dennoch stellt dieses Rauschen aus statistischer Sicht ein Problem dar. Es lässt sich mit klassischen Verfahren nicht durchdringen und zur Folge hat, dass sich die beobachteten erfahrungsbasierten Unterschiede zwischen den Gruppen annähern bzw. sich ausgleichen. Dies führt wiederum dazu, dass vorhandene Effekte eher abgeschwächt in Erscheinung treten.
Zusammenfassend kann festgehalten werden, dass sich diese beiden Faktoren limitierend auf die statistische Signifikanz der Ergebnisse in beiden Studien auswirkt. Hinsichtlich der deskriptiven Statistik empfiehlt es sich im Falle der durchgeführten Studien ein stärkeres Augenmerk auf den Median zu legen, welcher aufgrund der Bereinigungen im Vergleich zum Mittelwert reliablere Ergebnisse liefert (Bortz & Schuster, 2010). Ergänzend sei an dieser Stelle noch auf den Ausblick bei 8.3 in Kapitel 8 verwiesen. In diesem wird darauf eingegangen, wie zukünftig KI und auf ihr basierende Algorithmen genutzt werden können um tiefere Einblicke in Eye-Tracking-Datensätze zu erlangen.

7.2.4 Experten im Kontext dieser Dissertation

Im Kontext dieser Dissertation wird häufig zwischen Experten und Novizen unterschieden. Obwohl diese Begriffe aus Sicht der Expertiseforschung und dem von Erics-

son geschaffenen Verständnisse auf entsprechende Definitionen und eine fundierte empirische Untermauerung zurückgreifen können (Ericsson et al., 1993; Ericsson & Lehmann, 1996; Ericsson, 2006a; Ericsson et al., 2018; Ericsson & Towne, 2010; Gruber, 2007; Gruber, Lehtinen, Palonen & Degner, 2008; Gruber et al., 2010), weicht die Unterteilung in Gruppen in den zuvor beschriebenen Studien von diesen ab. Hier wurde nicht auf Basis des absolvierten *deliberate practice* aufgeteilt, sondern auf Grundlage der angesammelten *allgemeinen* und *professionellen Programmiererfahrung* (Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023). Diese Vorgehensweise geht darauf zurück, dass nicht ganz klar ist, wie sich *deliberate practice* bei Code Reviews definiert.
Eine frühere Arbeit von Hauser et al. (2020a) beschäftigt sich mit dieser Thematik und zeigt auf, dass von angehenden Entwicklern durchaus Aktivitäten durchgeführt werden, die dem Verständnis von Ericsson entsprechen (Ericsson et al., 1993; Ericsson & Towne, 2010). Allerdings fokussiert sich diese Arbeit vorrangig darauf, Aktivitäten zu identifizieren, die auf eine Leistungssteigerung abzielen und lässt aufgrund ihres qualitativen Charakters weitere Methoden der Expertiseforschung (z.B. Erfassung von Stunden, die mit Übungen verbracht werden) außer Acht (Hauser, Stark et al., 2020). Mehr Forschung zu diesem Thema ist für die Zukunft geplant.
Folglich sollte bedacht werden, dass es sich bei den Experten in dieser Studie um erfahrene Programmierer handelt, die jedoch nur bedingt die Kriterien der Expertiseforschung, wie beispielsweise die 10.000-Stunden-Regel erfüllen (Ericsson et al., 1993; Ericsson & Towne, 2010). In diesem Kontext und unter Berücksichtigung der der Ergebnisse von Hauser et al. (2020a) sollte eher die *expert performance* als relevanter Indikator betrachtet werden (Ericsson & Lehmann, 1996; Ericsson & Towne, 2010).

7.2.5 Methoden der Datenauswertung

Ebenfalls angesprochen werden sollen die verwendeten Methoden zur Datenauswertung in den beiden durchgeführten Studien.
Die allgemeine Betrachtung der Eye-Trcking-Daten, sowie deren Analyse findet sich in dieser Form auch in anderen Arbeiten in der Domäne des Software Engineerings oder der Erforschung von visueller Expertise wieder (Obaidellah et al., 2018; Sharafi et al., 2015). Hier gilt es erneut die häufig nicht signifikanten Ergebnisse anzusprechen, welche bereits erläutert worden sind.
Im Kontrast dazu, stellt die phasenbasierte Betrachtung einen neuen Schritt dar. Es wurde bereits in den entsprechenden Kapiteln 5 und 6 darauf hingewiesen, dass diese nicht unkritisch ist. Aufgrund der Unterteilung in drei Phasen ist diese etwas grob. Umso mehr Phasen erstellt und analysiert werden, umso genauer würden die Ergebnisse werden. Allerdings ist die Unterteilung des Datensatzes mit einem verhältnismäßig hohem Programmieraufwand in R verbunden. Dennoch bleibt diese Unterteilung ein Kritikpunkt.
Erneut sei an dieser Stelle auf den Ausblick auf zukünftige Studien und Methoden

(siehe Absatz 8.3) verwiesen. Um vorzugreifen: Der Einsatz von KI wird die Auswertung von Eye-Tracking-Daten vereinfachen und verlaufsorientierte Analysen der Daten ermöglichen, was eine deutlich feingranularere Betrachtung der angewandten Strategien erlauben wird (Ezer et al., 2023; Zemblys, Niehorster, Komogortsev & Holmqvist, 2018; Zemblys, Niehorster & Holmqvist, 2019).
Dennoch bleibt festzuhalten, dass die Unterteilung in drei Phasen aus testökonomischer Sicht den für diese Dissertation geeignetsten Schritt dargestellt hat.

7.3 Beantwortung der Forschungsfragen

Wie in Kapitel 4 beschrieben, verfolgt die hier vorliegende Dissertation insgesamt drei zentrale Ziele, die sich jeweils in entsprechende Unterfragen aufschlüsseln lassen. Wie aus den Studien 1 und 2 hervorgegangen ist, zeigen sich sowohl in C, als auch in C++ bestimmte Charakteristiken von visueller Expertise, die als programmiersprachenübergreifend betrachtet werden können. Insofern werden daher die in Kapitel 4 genannten Forschungsfragen zusammengefasst beantwortet. Dies findet sich im Anschluss in Absatz 7.3.1. Darauf folgend, soll die Thematik bezüglich des Aufbaus eines geeigneten Modells zur Interpretation von visueller Expertise bei Code Reviews abschließend adressiert werden. Hierzu finden sich Ausführungen, sowie die Darstellung eines eigenen Ansatzes in Absatz 7.3.2.

7.3.1 Beantwortung der Forschungsfragen zu visueller Expertise bei Code Reviews

In Kapitel 4 werden zwei Forschungsfragen genannt, welche sich sowohl für die Programmiersprache C, als auch C++ direkt auf die Erforschung visueller Expertise fokussieren. Diese sollen nachfolgend beantwortet werden.
Forschungsfrage 1 (*Wie unterscheiden sich Novizen und Experten hinsichtlich iherer Augenbewegungen während eines Code Reviews in den Programmiersprachen C/C++ im Allgemeinen?*) bezieht sich auf die allgemeine Betrachtung der relevanten Eye-Tracking-Metriken und begleitenden demografischen Daten. In beiden Studien zeigen sich Unterschiede zwischen den beiden Erfahrungsstufen. Sowohl bei C, als auch bei C++ zeigten die Experten bezüglich der Fehlererkennung eine bessere Leistung. Gleichzeitig deuteten die Eye-Tracking-Metriken darauf hin, dass sich die angesammelte *allgemeine* und *professionelle Programmiererfahrung* vorteilhaft auf diese im Sinne einer effizienteren Performance auswirken. Experten sind folglich in der Lage Code Reviews mit weniger visuellem Aufwand zu absolvieren und mehr Fehler zu identifizieren. Es bleibt jedoch zu beachten, dass sich die Unterschiede zwischen den Gruppen aufgrund der gewählten Datenanalyse und der Zusammenfassung im Sinne eines kontrastiven Vergleichs ausmitteln und zum Teil nur marginal ausfallen.
Forschungsfrage 2 (*Welche phasenbasierten Unterschiede machen sich in den Augenbewegungen von Novizen im Vergleich zu Experten bei der Durchführung eines Reviews in den Programmiersprachen C/C++ deutlich?* thematisiert die phasenbasierte Analyse der Da-

ten und welche Rückschlüsse diese auf visuelle Expertise erlaubt.

7.3.2 Beantwortung der Forschungsfragen zum Aufbau eines Modells zur Interpretation von visueller Expertise bei Code Reviews

Der zweite Fragenblock in Kapitel 4 widmet sich den *holistic models of image perception* und dem Aufbau eines vereinheitlichten Models zur Analyse und Interpretation von Augenbewegungen während eines Code Reviews.
Forschungsfrage 1 (*Welches der holistic models of image perception eignet sich am besten für die Analyse und Interpretation von Augenbewegungen während eines Code Reviews?*) und Forschungsfrage 2 (*Welche Komponenten der holistic models of image perception finden sich bei Code Reviews wieder?* haben einen direkten Bezug zu den *holistic models of image perception* und wurden bereits im Laufe des Kapitels diskutiert (siehe Absatz 7.1.2). So kann auf dieser Basis ausgesagt werden, dass sich das *global-focal search model* (Nodine & Kundel, 1987) und das *two-stage detection model* (Swensson, 1980) am besten für einen Einsatz bei Code Reviews eignen. Dies ist auf darauf zurückzuführen, dass die globale und fokale Betrachtung messbar sind und (vorgreifend auf Forschungsfrage 2) sich in dieser Form auch bei der Durchführung eines Code Reviews finden. Bezüglich der zweiten Forschungsfrage zeigt sich, dass sowohl die globale, als auch die fokale Betrachtung im Sinne des *global-focal search models* (Nodine & Kundel, 1987) in den Code Reviews auftauchen. Ebenso kann festgehalten werden, dass die stärkere Betonung und die Filterfunktion des *two-stage detection models* von Swensson (1980) auf Seiten der Experten zu einer besseren Fehlererkennung und effizienteren Durchführung des Reviews führen. Für den von Kundel et al. (2007) beschriebenen Ansatz des *holistic mode vs. search to find* gibt es zwar Anhaltspunkte, jedoch bedürfen diese hinsichtlich einer empirischen Verifizierung einer tiefergehenden Methodik und der Erfassung weiterer Eye-Tracking-Metriken.
Forschungsfrage 3 (*Wie müsste ein vereinheitlichtes Gesamtmodel zur Analyse und Interpretation von Augenbewegungen während eines Code Reviews aussehen?*) adressiert die bei Absatz 7.1.2 genannten Schwachstellen der *holistic models of image perception* und konzentriert sich auf die Formulierung eines vereinheitlichten Modells. Bei genauerer Betrachtung dieser Frage und unter Berücksichtigung der erlangten Ergebnisse zeigt sich, dass diese nicht kurz beantwortet werden kann. Dieser Forschungsfrage widmet sich der nachfolgende Absatz welcher die Formulierung eines vereinheitlichten Modells zum Thema hat (siehe Absatz 7.4).

7.4 Das vereinheitlichte Modell zur Analyse und Interpretation von Augenbewegungen während eines Code Reviews

Basierend auf den aus der C- und C++-Studie erlangten Ergebnissen, der Einbeziehung der Erkenntnisse aus der Expertiseforschung und dem Abgleich mit den Resultaten von anderen Forschern soll nachfolgend die dritte Forschungsfrage zu den

holistic models of image perception adressiert werden. Diese fokussiert vorrangig den Aufbau eines vereinheitlichten Gesamtmodels für die Anwendung auf Code Reviews .
Wie bereits zuvor dargelegt wurde, müsste sich ein vereinheitlichtes Modell zur Interpretation von Augenbewegungen während eines Code Reviews vorrangig auf das *global-focal search model* (Nodine & Kundel, 1987) stützen und das von Swensson (1980) thematisierte und auch in der Expertiseforschung relevante Vorwissen der Versuchspersonen einbeziehen. Der Entwurf eines entsprechenden Modells wird in Abbildung 7.5 dargestellt und soll nachfolgend beschrieben werden.

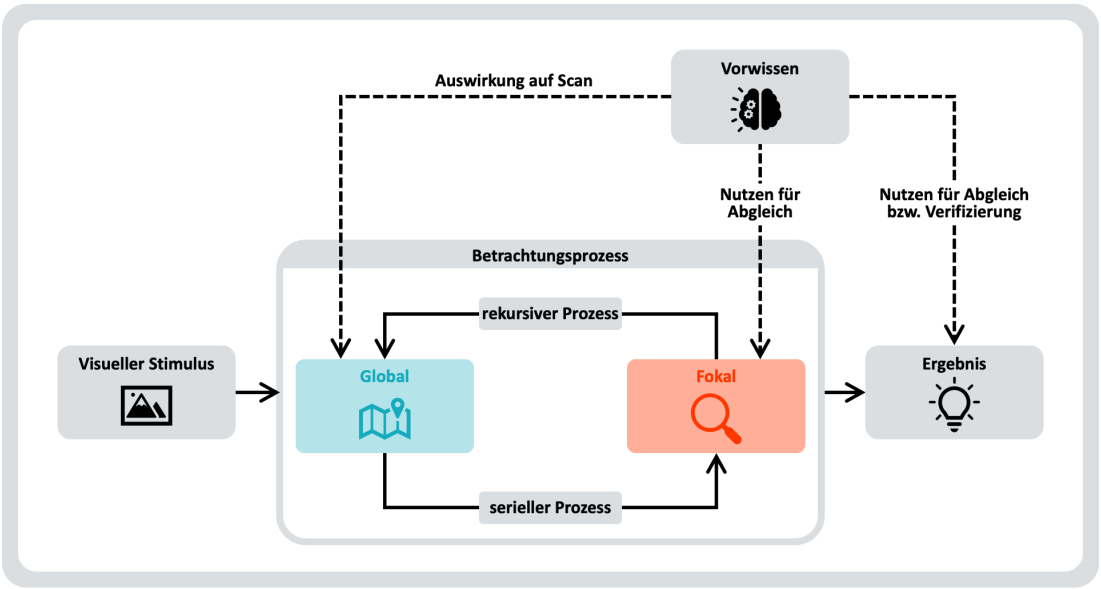

Abbildung 7.5: Schematische Darstellung des *unified holistic model of visual perception*

Das in Abbildung 7.5 präsentierte Modell vereint die Charakteristiken des *global-focal search model* (Nodine & Kundel, 1987) und des *two-stage detection model* (Swensson, 1980) und setzt diese in Bezug zu den Annahmen der Expertiseforschung (Ericsson, 2016; Ericsson & Towne, 2010; Gegenfurtner et al., 2011; Gruber, 2007). Insbesondere die *information-reduction hypothesis* (Haider & Frensch, 1996, 1999) und die *theory of long-term working memory* (Ericsson & Kintsch, 1995) werden im Kontext des Modells und hinsichtlich ihres Einflusses auf die Betrachtungsprozesse berücksichtigt. Als Arbeitstitel trägt es *unified holistic model of visual perception*.
Das *unified holistic model of visual perception* gleicht in seinem Aufbau dem bereits genannten *global-focal search model*, bezieht jedoch das individuelle und domänenspezifische Vorwissen deutlich stärker in den Betrachtungsprozess ein. Die Rolle des Vorwissens erfüllt zwar die von Swensson (1980) beschriebene Filterfunktion, geht jedoch über diese hinaus bzw. wird durch weitere Funktionen ergänzt. Im Sinne der *information-reduction hypothesis* (Haider & Frensch, 1996, 1996), der *theory of long-term working memory* (Ericsson & Kintsch, 1995), sowie der Expertiseforschung im Allgemeinen (Ericsson, 2006a, 2016; Ericsson et al., 1993; Ericsson & Towne, 2010; Gegenfurtner, 2020; Gegenfurtner et al., 2011; Gruber, 2007; Gruber & Lehmann, 2008) ist davon auszugehen, dass das Vorwissen eines Experten bezüglich der Durchführung

eines Code Reviews eine umfangreiche Ansammlung von typischen Fallbeispielen bereithält. Aufgrund der Strukturierung des Langzeitgedächtnisses sind diese für Experten leicht abrufbar und auf das jeweils aktuelle Review übertragbar (Ericsson & Kintsch, 1995; Gruber, 2007). Ebenso können Experten bei einem Review deutlich besser zwischen anfälligen Stellen und eher robusten Abschnitten des Quellcodes unterscheiden und folglich ihre Aufmerksamkeit im Sinne der *information-reduction hypothesis* (Haider & Frensch, 1996, 1999; Kok, 2016) besser steuern. Werden Areale bereits beim ersten globalen Durchlauf als anfällig betrachtet, erhalten sie mehr Aufmerksamkeit, wohingegen unkritische Stellen vernachlässigt werden. Gleichzeitig wirkt sich das Vorwissen in Form einer Filterfunktion bereits auf die initiale Durchführung eines Scans aus und hilft dem Reviewer den vorliegenden Code zu klassifizieren. Ergänzend dazu, ist davon auszugehen, dass die Experten über ein breit ausgebautes Wissen über Fälle und prototypische Beispiele verfügen, welche im Langzeitgedächtnis abgespeichert sind und bei Bedarf abgerufen werden können (Ericsson & Kintsch, 1995). Gleichzeitig kann davon ausgegangen werden, dass das Vorwissen den gesamten Betrachtungsprozess bzw. das gesamte Review unterstützt und immer wieder für einen Abgleich des vorliegenden Codes mit bereits gesammelten Erfahrungen genutzt wird. Somit ist es in jedem Abschnitt der Begutachtung präsent und übt einen nicht zu unterschätzenden Einfluss auf die gezeigten Augenbewegungen aus.

Der Betrachtungsprozess des *unified holistic model of visual perception* orientiert sich stark an dem von Nodine und Kundel (1987) beschriebenen *global-focal search model*. Er unterteilt sich ebenfalls in globale und fokale Phasen. Wie im Modell von Nodine und Kundel (1987) wird davon ausgegangen, dass sich die beiden Phasen während des Reviews mehrfach abwechseln und diese solange durchlaufen werden, bis eine zufriedenstellende Lösung erzielt werden kann (Begel & Vrzakova, 2018). Der Einfluss des Vorwissens wirkt sich wie im vorangegangenen Absatz beschrieben auf beide Phasen aus: Die globale Betachtung wird aufgrund des Vorwissens bereits dahingehend beeinflusst, dass vulnerable Stellen schneller identifiziert werden können (Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006). Die in der fokalen Phase zu analysierenden Auffälligkeiten werden mit dem individuellen und domänensepzifischen Vorwissen (beispielsweise ähnliche Fälle, Wissen über programmiersprachentypsiche Fehler, ...) abgeglichen (Ericsson & Kintsch, 1995).

Das erzielte Ergebnis des Betrachtungsprozesses wird ebenfalls mit dem Vorwissen abgeglichen und auf Basis der domänenspezifischen Erfahrungen validiert (Ericsson & Kintsch, 1995; Nodine & Kundel, 1987). Im Falle der Code Reviews konnte beobachtet werden, dass die abgegebenen Fehlerbeschreibungen von Experten einen größeren Nutzen für andere Reviewer darstellen würden (Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Nivala et al., 2016). Diese sind auf eine Weise verfasst, die den identifizierten Fehler genau beschreiben, auf dessen Auswirkungen und Verflechtungen im Quellcode eingehen und einen Lösungsansatz präsentieren.

Zusammenfassend stellt das hier erstellte und auf Code Reviews angewandte *unified holistic model of visual perception* eine Synthese aus den *holistic models of image perception* und der Expertiseforschung dar (Ericsson & Kintsch, 1995; Ericsson et al., 1993; Gruber, 2007; Haider & Frensch, 1996, 1999; Kundel et al., 2007; Nodine & Kundel, 1987; Sheridan & Reingold, 2017; Swensson, 1980). Im hier präsentierten Stadium bietet das Modell eine Grundlage für die Erfassung und Interpretation von visueller Expertise bei der Suche von Fehlern in Quellcodes, ist jedoch nicht auf diese limitiert. Im Laufe weiterer Studien soll es validiert und auf verschiedene Themenbereiche (beispielsweise UML im Software Engineering) angewandt werden. Ähnliche Synthesen haben sich auch schon in andere Arbeiten gefunden (Gegenfurtner, 2020; Kok, 2016; Sheridan & Reingold, 2017) und sollten hinsichtlich zukünftiger Modifikationen am *unified holistic model of visual perception* einbezogen werden.

Kapitel 8

Schlussfolgerungen

Dieses Kapitel widmet sich den Schlussfolgerungen, welche aus dieser Dissertation gezogen werden können. Auf den folgenden Seiten soll zusammengefasst werden, welchen Beitrag die Arbeit hinsichtlich der Erforschung von visueller Expertise bei Code Reviews leistet und welche Impulse daraus für neue Studien gezogen werden können (siehe Absatz 8.1). Weiterhin soll auf die Bedeutung von phasenbasierten Analysen von Eye-Tracking-Daten im Kontext der Erforschung visueller Expertise im Allgemeinen eingegangen werden (siehe Absatz 8.2). Den Abschluss stellt ein Ausblick auf mögliche zukünftige Veränderungen in dieser Domäne dar (siehe Absatz 8.3).

8.1 Visuelle Expertise bei Code Reviews

Obwohl durch die vorangegangene kritische Würdigung der durchgeführten Studien und den erlangten Ergebnissen dieser Dissertation klar wird, dass sie keinen Anspruch auf eine umfassende Lösung bzw. die Beschreibung von visueller Expertise bei Code Reviews erhebt (siehe Absatz 7.2), leistet sie dennoch einen Beitrag um diesen Forschungsbereich voranzutreiben. So zeigt sie deutlich, dass die Verwendung der *holistic models of image perception* zur Analyse und Interpretation von Augenbewegungen während eines Code Reviews einen geeigneten Ansatz darstellen und liefert mit dem in Absatz 7.4 beschriebenen *unified holistic model of visual perception* einen erweiterten Lösungsansatz, der in zukünftigen Studien als Grundlage genutzt werden kann.
Ergänzend dazu leistet die Disertation durch die Verwendung der genannten Metriken der *holistic models of image perception*, sowie unter Einbeziehung der Expertiseforschung im Sinne von Ericsson (Ericsson, 2016; Ericsson et al., 2018, 1993; Ericsson & Towne, 2010) einen Beitrag zur Standardisierung von Eye-Tracking-Studien im Software Engineering und zeigt, dass auch Ansätze aus anderen Domänen genutzt werden können. Damit adressiert sie die von Obaidellah et al. (2018) und Sharafi et al. (2015) in ihren Metastudien kritisierte (unreflektierte) Verwendung von Theorien und Ansätzen aus anderen Domänen in Bezug auf die Analyse von Augenbewegungen im Software Engineering.

Werden zukünftig ähnliche Studien zu Code Reviews durchgeführt, so ergeben sich aus dieser Dissertation ebenfalls Vorschläge: Es sollte der Bedeutung von globaler und fokaler Betrachtung mehr Aufmerksamkeit geschenkt werden. Ebenso stellen die Wechsel zwischen diesen Phasen einen interessanten Forschungsbereich dar. Zukünftige Arbeiten könnten der Frage nachgehen, wann und wieso die Betrachtungsweise gewechselt wird. Diesbezüglich empfehlen sich triangulative Designs und der Einsatz von KI-Algorithmen zur Identifizierung der relevanten Phasen. Ebenso stellen andere Themenbereiche des Software Engineerings (beispielsweise die UML-Modellierung oder das Requirement Engineering) Bereiche dar, in welchen visuelle Strategien zwar von großer Bedeutung sind, jedoch zum aktuellen Zeit noch nicht erforscht wurden (Hauser et al., 2019; Hutzler et al., 2018; Obaidellah et al., 2018; Sharafi et al., 2015; Sommerville, 2012).

Auch sollte zukünftig thematisiert werden, wie sich die gewonnenen Erkenntnisse dieser Dissertation bzw. das Expertenwissen zu Code Reviews im Allgemeinen transferieren und für Novizen greifbar machen lässt. Ein probates Mittel könnten in diesem Zuge *Eye Movement Modeling Examples* (Hauser, Grabinger, Mottok, Jahn & Nadimpalli, 2023; Jarodzka et al., 2010; Stark, 2021). Erste Schritt in diese Richtung wurde bereits 2020 und 2023 unternommen, indem für die Lehrveranstaltungen zum Software Engineering an der OTH Regensburg *Eye Movement Modeling Examples* experimentell untersucht (Stark, 2021) und in die Lehre integriert wurden (Hauser, Grabinger, Mottok, Jahn & Nadimpalli, 2023). Diese spezielle Form von Lernvideos hat das Potenzial die im Rahmen dieser Dissertation beschriebenen Strategien der Experten für Novizen greifbarer zu machen.

Basierend auf den Erkenntnissen dieser Dissertation kann bezüglich visueller Expertise bei Code Reviews abschließend ausgesagt werden, dass sich zwischen Experten und Novizen Unterschiede in ihren Vorgehensweisen finden. Diese sind auf die angesammelte domänenspezifische Erfahrung zurückzuführen. Die Unterschiede zeigen sich sowohl in der gemessenen Leistung, als auch in den Augenbewegungen, welche im Verlauf des Expertiseerwerbs optimiert werden und durch ein phasenbasiertes Vorgehen gekennzeichnet sind (dieses wird im nachfolgenden Absatz 8.2 auch über Code Reviews hinaus thematisiert).

Zusammenfassend kann festgehalten werden, dass sich visuelle Expertise bei Code Reviews im Sinne der in Kapitel 2 bei Absatz 2.3 genannten Arbeitsdefinition als eine domänenspezfische Kompetenz auswirkt, welche im Laufe eines jahrelangen Trainings aufgebaut wird (Hauser, Stark et al., 2020). Sie geht einher mit der Anhäufung von fachspezifischen Wissen (z.B. über den Aufbau von Quellcodes, programmiersprachenspezifische Fehler, ...) und der Anpassung bzw. Optimierung der eigenen kognitiven Strategien, welche sich in den Augenbewegungen bemerkbar machen und eine effizientere Durchführung des Reviews ermöglichen (Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Gegenfurtner et al., 2017; Gegenfurtner & Merriënboer, 2017; Kok, 2016).

Abschließend bleibt speziell zur visuellen Expertise bei Code Reviews zu sagen, dass

sie im Zuge der stark zunehmenden Bedeutung von Software für unsere Gesellschaft und unsere Infrastruktur (Broy, 2006; Sommerville, 2012) einen interessanten Themenbereich darstellt, der zunehmend mehr in den Fokus der Forschung gerät (siehe diesbezüglich Absatz 3.2.4) und von den technischen Entwicklungen hinsichtlich des Eye-Trackings profitiert (Obaidellah et al., 2018; Sharafi et al., 2015).

8.2 Phasenbasierte Analyse von Augenbewegungen

Die in den vorangegangenen Kapiteln präsentierten Studien, sowie die Publikationen zahlreicher weiterer Forscherinnen und Forscher legen nahe, dass Augenbewegungen bei Code Reviews phasenbasiert ablaufen (Begel & Vrzakova, 2018; Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023; Nivala et al., 2016; Obaidellah et al., 2018; Sharafi et al., 2015; Sharif et al., 2012; Turner et al., 2014; Uwano et al., 2006). Obwohl diese Annahme bereits seit längerer Zeit existiert, wurde sie erstmalig im Kontext dieser Dissertation ausführlicher verfolgt. Bei den Recherchearbeiten, welche in Kapitel 3 und zum Teil auch in Kapitel 2 geschildert werden, hat sich gezeigt, dass diese phasenbasierten Ansätze zum aktuellen Zeitpunkt eher unterschätzt werden. So finden sich entsprechende Studien vorrangig in Bezug auf die Analyse und Interpretation von medizinischen Bildmaterialien (Gegenfurtner et al., 2011, 2017; Kok, 2016; Sheridan & Reingold, 2017), erfahren aber außerhalb dieser Domäne wenig Nutzung. Dies mag mehrere Gründe haben.
Wie bereits geschildert wurde, stellt sich die Datenaufbereitung für die phasenbasierte Betrachtung als relativ komplex und arbeitsintensiv dar. Dies hat zur Folge, dass diese Vorgehensweise nicht für jede Studie geeignet ist und bestimmte Kenntnisse erfordert. So setzte beispielsweise die in den Kapiteln 5 und 6 gezeigte Datenaufbereitung zwingend ein Arbeiten mit den Rohdaten des jeweiligen Eye-Trackers voraus (Sharafi et al., 2020), was wiederum (zumindest grundlegende) Programmierkenntnisse in einer entsprechenden Sprache (beispielsweise R oder Python) bedingt.
Ebenso könnte der übergreifende Name der *holistic models of image perception* ein Problem darstellen. Diesen Modellen liegt de facto ein phasenbasiertes Vorgehen hinsichtlich der Betrachtung eines visuellen Stimulus zugrunde, jedoch impliziert das Wort *image* in diesem Fall eine Anwendung auf Bildmaterial. Wie sich im Fall der Code Reviews gezeigt hat, können diese Modelle allerdings ein deutlich breiteres Spektrum abdecken und Ergebnisse produzieren. Dennoch wurde beim im Absatz 7.4 dargestellten *unified holistic model of visual perception* der Titel bewusst so gewählt, dass sich dieser nicht limitierend auf die möglichen Anwendungen auswirkt.
Insgesamt sollte vor allem in Bezug auf die Erforschung von visueller Expertise in zukünftigen Studien überlegt werden, ob die zeitlichen Veränderungen der Eye-Tracking-Metriken während der Begutachtung eines Stimulus nicht stärker gewichtet werden. Dieser Ansatz bietet Forscherinnen und Forschern die Möglichkeit ein tieferes Verständnis für die involvierten Strategien zu erlangen und beobachtete Unterschiede hinsichtlich des Expertisegrades zu interpretieren.

8.3 Zukünftige Veränderungen der Domäne

Die Erforschung von visueller Expertise bei Code Reviews bzw. übergreifend im Software Engineering wird in den nächsten Jahren durch mehrere technologische Entwicklungen geprägt werden. Von großer Bedeutung wird dabei sein, dass Augenbewegungen in natürlichen, möglichst realistischen Szenarien erfasst werden können (siehe Absatz 8.3.1) und die Datenanalyse durch die Verwendung von KI unterstützt wird (siehe Absatz 8.3.2). Beide Aspekte werden nachfolgend angesprochen.

8.3.1 Eye-Tracking in der IDE

Die in Absatz 7.2.1 angesprochene Verwendung von künstlich erstellten Codebeispielen und der Einsatz einer speziellen Experimentalsoftware wie beispielsweise Tobii ProLab (Tobii Pro, 2021) oder SMI BeGaze (SensoMotoric Instruments, 2017) wirkte sich hinsichtlich der Untersuchung von visueller Expertise bei Code Reviews relativ limitierend aus. Abhilfe könnte an dieser Stelle beispielsweise durch die Verwendung von neuen Softwareprodukten geschaffen werden.
So entwickelte beispielsweise die Forschungsgruppe um Bonita Sharif die Software iTrace (Clark & Sharif, 2017; Guarnera, Bryant, Mishra, Maletic & Sharif, 2018; Shaffer et al., 2015; Sharif et al., 2016), welche in IDEs integriert werden kann und so ein natürlicheres Arbeiten mit Quellcode bei Eye-Tracking-Studien erlaubt. Gleichzeitig bietet iTrace auch eine Scrollkompensation, welche die Verwendung längerer Codebeispiele oder die Nutzung von echten Programmen aus verschiedenen Kontexten erlaubt.
Die Software wurde Ende 2023 im Regensburger Eye-Tracking-Classroom erstmals im Rahmen einer gemeinsamen Studie mit Bonita Sharif und Jonathan Maletic genutzt und ist seitdem Bestandteil der Softwareausstattung. Auf Basis einer bestehende Kooperation mit Bonita Sharif wird iTrace für die Durchführung von weiteren Studien am Standort Regensburg häufiger zum Einsatz kommen.

8.3.2 Phasenbasierte Betrachtung von Eye-Tracking-Metriken mittels KI

Obwohl KI noch vor einigen Jahren nicht für die breite Masse zugänglich war bzw. sich deren Nutzen eher in Grenzen hielt, kommt dieser spätestens mit der Veröffentlichung von ChatGPT im Juni 2020 und deren aktuell zunehmender Beliebtheit verstärktes Interesse zu (OpenAI, 2024). In der Eye-Tracking-Forschung fanden sich bereits vor 2020 erste Nutzungsansätze von KI (Zemblys et al., 2018, 2019) und es lässt sich beobachten, dass diese Technologie zunehmend mehr für Datenauswertungen von Studien genutzt wird (Bittner et al., 2023; Ezer et al., 2023). KI wird den Forscherinnen und Forschern neue Möglichkeiten in Bezug auf die Auswertung von Blickbewegungsdaten ermöglichen (Ezer et al., 2023; Haselhuhn, 2020; Zemblys et al., 2018, 2019).
So könnten beispielsweise die in Studie 1 und 2 beschriebenen phasenbasierten Analysen zum aktuellen Zeitpunkt mit entsprechenden KI-Algorithmen deutlich fein-

granularer durchgeführt werden (Hauser, Reuter et al., 2018; Hauser, Schreistetter et al., 2020; Hauser, Grabinger, Mottok & Gruber, 2023). Eine KI könnte darauf trainiert werden, *global* und *focal viewing* (Nodine & Kundel, 1987) zu erkennen und darauf basierend den gesamten Verlauf eines Reviews betrachten. Diese Vorgehensweise würde eine deutlich präzisere Betrachtung der Veränderungen der relevanten Metriken ermöglichen und diese von zeitlichen Einteilungen, wie beispielsweise der hier verwendeten Drittelung in Phasen loslösen.
Ebenso sollte nicht unterschätzt werden, dass KI auch in anderen Bereichen, als nur in der Datenauswertung genutzt werden kann. Es wäre denkbar, dass diese in Form von adaptiven Supportsystemen eingesetzt wird. Reviewprozesse könnten mit einer Kombination aus Eye-Tracking und KI unterstützt werden. Die Anwendungsbeispiele sind in diesem Fall breit gestreut: Von einer simplen Müdigkeitserkennung bis hin zu adaptiven Feedbacksystemen ist eine Nutzung vorstellbar.

Literaturverzeichnis

Abid, N. J., Maletic, J. I. & Sharif, B. (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 '19* (S. 1–9). Denver, CO, USA: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3314111.3319834 doi: 10.1145/3314111.3319834

Abran, A., Moore, J. W., Bourque, P. & Dupuis, R. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. doi: 10.1234/12345678

Adam Just, M. & Carpenter, P. A. (1984). Using Eye Fixations to Study Reading Comprehension. In *New methods in reading comprehension research* (S. 151–182). Routledge. Zugriff auf https://www.taylorfrancis.com/books/9780429999710/chapters/10.4324/9780429505379-8 doi: 10.4324/9780429505379-8

Alzubaidi, M., Black, J. A., Patel, A. & Panchanathan, S. (2009). Conscious vs. subconscious perception, as a function of radiological expertise. *Proceedings - IEEE Symposium on Computer-Based Medical Systems*. doi: 10.1109/CBMS.2009.5255353

Antes, J. R. & Kristjanson, A. F. (1991). Discriminating Artists from Nonartists by Their Eye-Fixation Patterns. *Perceptual and Motor Skills*, *73* (3), 893–894. Zugriff auf http://journals.sagepub.com/doi/10.2466/pms.1991.73.3.893 doi: 10.2466/pms.1991.73.3.893

Aschwanden, C. & Crosby, M. (2006). Code Scanning Patterns in Program Comprehension. In *Proceedings of the 39th hawaii international conference on system sciences*. Kauia, HI, USA. Zugriff auf http://pdf.aminer.org/000/641/141/is_there_any_difference_in_novice_comprehension_of_a_small.pdf

Assaf, D., Amar, E., Marwan, N., Neuman, Y., Salai, M. & Rath, E. (2016). Dynamic patterns of expertise: The case of orthopedic medical diagnosis. *PLoS ONE*, *11* (7), 1–12. doi: 10.1371/journal.pone.0158820

Atal, R. & Sureka, A. (2015). Anukarna: A software engineering simulation game for teaching practical decision making in peer code review. In *Proceedings of the 1st international workshop on case method for computing education (cmce)* (S. 63–70).

Bacchelli, A. & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 35th ieee international conference on software engineering (icse)* (S. 712–721). San Francisco, CA, USA: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/6606617/ doi: 10.1109/

ICSE.2013.6606617

Badampudi, D., Britto, R. & Unterkalmsteiner, M. (2019). Modern code reviews - Preliminary results of a systematic mapping study. *ACM International Conference Proceeding Series*, 340–345. doi: 10.1145/3319008.3319354

Baker, R. A. (1997). Code reviews enhance software quality. *Proceedings - International Conference on Software Engineering*, 570–571. doi: 10.1145/253228.253461

Banks, V. A., Plant, K. L. & Stanton, N. A. (2018). Driver error or designer error: Using the Perceptual Cycle Model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016. *Safety Science*, *108*, 278–285. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0925753517314212 doi: 10.1016/j.ssci.2017.12.023

Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E. & Parnin, C. (2017). Do Developers Read Compiler Error Messages? In *2017 ieee/acm 39th international conference on software engineering (icse)* (S. 575–585). Buenos Aires, Argentinien: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/7985695/ doi: 10.1109/ICSE.2017.59

Baum, T., Leßmann, H. & Schneider, K. (2017). The Choice of Code Review Process: A Survey on the State of the Practice. In M. Felderer, D. M. Fernández, B. Turham, M. Kalinowski, F. Sarro & D. Winkler (Hrsg.), *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (S. 111–127). Cham, Schweiz: Springer International Publishing. Zugriff auf http://link.springer.com/10.1007/978-3-319-69926-4_17 doi: 10.1007/978-3-319-69926-4_17

Baum, T., Liskin, O., Niklas, K. & Schneider, K. (2016). Factors influencing code review processes in industry. In *Proceedings of the 2016 24th acm sigsoft international symposium on foundations of software engineering - fse 2016* (S. 85–96). New York, USA: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=2950290.2950323 doi: 10.1145/2950290.2950323

Bayerisches Staatsministerium für Unterricht und Kultus. (2020). *Stiftung Bildungspaket Bayern: Digitale Schule 2020.* https://bildungspakt-bayern.de/digitale-schule-2020/. Zugriff am 2020-02-12 auf https://bildungspakt-bayern.de/digitale-schule-2020/ (Accessed: 2020-02-12)

Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie. (2020). *Digitalbonus Bayern.* https://www.stmwi.bayern.de/service/foerderprogramme/digitalbonus-bayern/. Zugriff am 2020-02-12 auf https://www.stmwi.bayern.de/service/foerderprogramme/digitalbonus-bayern/ (Accessed: 2020-02-12)

Bednarik, R. (2012). Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human Computer Studies*, *70* (2), 143–155. Zugriff auf http://dx.doi.org/10.1016/j.ijhcs.2011.09.003 doi: 10.1016/j.ijhcs.2011.09.003

Bednarik, R., Schulte, C., Budde, L., Heinemann, B. & Vrzakova, H. (2018). Eye-movement Modeling Examples in Source Code Comprehension. In *Koli calling international conference on computing education research (koli calling '18)* (S. 1–8). Koli, Finnland: ACM Inc. Zugriff auf http://dl.acm.org/citation.cfm?doid=3279720.3279722 doi: 10.1145/3279720.3279722

Bednarik, R. & Tukiainen, M. (2006). An eye-tracking methodology for characterizing program comprehension processes. In *Eye tracking research applications (etra)* (S. 125–132). San Diego, CA, USA: ACM. doi: 10.1145/1117309.1117356

Bednarik, R. & Tukiainen, M. (2007). Validating the Restricted Focus Viewer: a study using eye-movement tracking. *Behavior research methods*, *39* (2), 274–282. doi: 10.3758/BF03193158

Bednarik, R. & Tukiainen, M. (2008). Temporal eye-tracking data: Evolution of Debugging Strategies with Multiple Representations. In *Proceedings of the symposium on eye tracking research and applications (etra)* (Bd. 1, S. 99–102). Savannah, GA, USA: ACM. doi: 10.1145/1344471.1344497

Beelders, T. & du Plessis, J.-P. (2016). The Influence of Syntax Highlighting on Scanning and Reading Behaviour for Source Code. In *Proceedings of the annual conference of the south african institute of computer scientists and information technologists (saicsit)* (S. 1–10). Johannesburg, Südafrika: ACM. doi: 10.1145/2987491.2987536

Begel, A. & Vrzakova, H. (2018). Eye movements in code review. In *Proceedings of the workshop on eye movements in programming (emip)* (S. 1–5). Warschau, Polen: ACM. doi: 10.1145/3216723.3216727

Beller, M., Bacchelli, A., Zaidman, A. & Juergens, E. (2014). Modern code reviews in open-source projects: Which problems do they fix? *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 202–211. doi: 10.1145/2597073.2597082

Bergstrom, J. R. & Schall, A. (2014). *Overview of eye tracking and visual search*. Amsterdam, Niederlande: Elsevier Science Technology.

Bergua, A. (2017). *Das menschliche Auge in Zahlen*. Berlin, Deutschland: Springer. Zugriff auf http://link.springer.com/10.1007/978-3-662-47284-2 doi: 10.1007/978-3-662-47284-2

Berliner, D. C. (2001). Learning about and learning from expert teachers. *International Journal of Educational Research*, *35* (5), 463–482. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0883035502000046 doi: 10.1016/S0883-0355(02)00004-6

Biddle, R., Noble, J. & Tempero, E. (2003). Teaching the Evaluation of Object-Oriented Designs. *Fifth Australasian Computing Education Conference (ACE2003)*, *20*, 213–220. Zugriff auf http://crpit.com/confpapers/CRPITV20Biddle.pdf

Billett, S., Harteis, C. & Gruber, H. (2018). Developing occupational expertise through everyday work activities and interactions. In K. A. Ericsson, R. R. Hoffman, A. Kozbelt & A. M. Williams (Hrsg.), *The camebridge handbook of expertise and*

expert performance (2nd Aufl., S. 105–126). Cambridge, Vereinigtes Königreich: Cambridge University Press.

Binkley, D., Davis, M., Lawrie, D., Maletic, J. I., Morrell, C. & Sharif, B. (2013). *The impact of identifier style on effort and comprehension* (Bd. 18) (Nr. 2). doi: 10.1007/s10664-012-9201-4

Bittner, D., Ezer, T., Grabinger, L., Hauser, F. & Mottok, J. (2023). Unveiling the Secrets of Learning Styles: Decoding Eye Movements Via Machine Learning. In *Proceedings of the 16th annual international conference of education, research and innovation* (S. 5153–5162). Sevilla, Spanien: IATED Academy. doi: 10.21125/iceri.2023.1291

Boehm, B. & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *Software Management*, *34* (1), 135–137. Zugriff auf http://ieeexplore.ieee.org/document/962984/ doi: 10.1109/2.962984

Bortz, J. & Schuster, C. (2010). *Statistik für Human- und Sozialwissenschaftler*. Berlin, Heidelberg: Springer Berlin Heidelberg. Zugriff auf http://link.springer.com/10.1007/978-3-642-12770-0 doi: 10.1007/978-3-642-12770-0

Bosu, A., Greiler, M. & Bird, C. (2015). Characteristics of useful code reviews: An empirical study at Microsoft. *IEEE International Working Conference on Mining Software Repositories*, *2015-Augus*, 146–156. doi: 10.1109/MSR.2015.21

Boyini, K. (2019). *What is Memory Leak in C / C ++ ? Output Example.* Zugriff auf https://www.tutorialspoint.com/what-is-memory-leak-in-c-cplusplus

Broy, M. (2006). Challenges in automotive software engineering. In *Proceedings of the acm international conference on software engineering (icse)* (S. 33–42). Shanghai, China: ACM.

Brunyé, T. T., Carney, P. A., Allison, K. H., Shapiro, L. G., Weaver, D. L. & Elmore, J. G. (2014). Eye movements as an index of pathologist visual expertise: A pilot study. *PLoS ONE*, *9*. doi: 10.1371/journal.pone.0103447

Bühner, M. & Ziegler, M. (2017a). Mediation und Moderation in SPSS und R. In *Statistik für psychologen und sozialwissenschaftler grundlagen und umsetzung mit spss und r* (Kap. 7.3). Hallbergmoos: Pear.

Bühner, M. & Ziegler, M. (2017b). Multiple lineare Regression.pdf. In *Statistik für psychologen und sozialwissenschaftler grundlagen und umsetzung mit spss und r* (Kap. 7.2). Hallbergmoo.

Buse, R. P. L. & Weimer, W. R. (2010). Learning a Metric for Code Readability. *IEEE Transactions on Software Engineering*, *36* (4), 546–558. Zugriff auf http://ieeexplore.ieee.org/document/5332232/ doi: 10.1109/TSE.2009.70

Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J. H., Schulte, C., … Tamm, S. (2015). Eye Movements in Code Reading: Relaxing the Linear Order. In *Proceedings of the 23rd ieee international conference on program comprehension (icpc)* (S. 255–265). IEEE. doi: 10.1109/ICPC.2015.36

Busjahn, T., Bednarik, R. & Schulte, C. (2014). What influences dwell time during

source code reading? In *Proceedings of the acm symposium on eye tracking research applications (etra)* (S. 335–338). Safety Harbor, FL, USA: ACM. doi: 10.1145/2578153.2578211

Busjahn, T., Schulte, C. & Busjahn, A. (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 '11* (S. 1). New York, NY, USA: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=2094131.2094133 doi: 10.1145/2094131.2094133

Busjahn, T., Shchekotova, G., Antropova, M., Schulte, C., Sharif, B., Simon, ... Ihantola, P. (2014). Eye tracking in computing education. In *Proceedings of the tenth annual conference on international computing education research - icer '14* (S. 3–10). New York, NY, USA: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=2632320.2632344 doi: 10.1145/2632320.2632344

Canham, M. & Hegarty, M. (2010). Effects of knowledge and display design on comprehension of complex graphics. *Learning and Instruction*, *20* (2), 155–166. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0959475209000218 doi: 10.1016/j.learninstruc.2009.02.014

Carullo, G. (2020). *Implementing Effective Code Reviews*. Berkeley, CA, USA: Apress. Zugriff auf http://link.springer.com/10.1007/978-1-4842-6162-0 doi: 10.1007/978-1-4842-6162-0

Cha, S., Taylor, R. N. & Kang, K. (2019). *Handbook of Software Engineering* (S. Cha, R. N. Taylor & K. Kang, Hrsg.). Cham, Schweiz: Springer International Publishing. Zugriff auf http://link.springer.com/10.1007/978-3-030-00262-6 doi: 10.1007/978-3-030-00262-6

Chandrika, K. R., Amudha, J. & Sudarsan, S. D. (2018). Recognizing eye tracking traits for source code review. In *Ieee international conference on emerging technologies and factory automation, etfa* (S. 1–8). Torina, Italien: IEEE. doi: 10.1109/ETFA.2017.8247637

Chappell, B. & Gonzales, R. (2019). *FAA To Order Changes In Boeing 737 Max Jets After Ethiopian Airlines Crash.* Zugriff auf https://www.npr.org/2019/03/11/702150411/black-box-recorders-found-in-ethiopian-airlines-crash-of-boeing-737-max-8-jet

Charness, N., Reingold, E. M., Pomplun, M. & Stampe, D. M. (2001). The perceptual aspect of skilled performance in chess: Evidence from eye movements. *Memory Cognition*, *29* (8), 1146–1152. Zugriff auf http://link.springer.com/10.3758/BF03206384 doi: 10.3758/BF03206384

Charness, N., Tuffiash, M., Krampe, R., Reingold, E. & Vasyukova, E. (2005). The role of deliberate practice in chess expertise. *Applied Cognitive Psychology*, *19* (2), 151–165. doi: 10.1002/acp.1106

Chi, M. T. H. (2006). Methods to assess the representations of experts' and novices' Knowledge. In *Cambridge handbook of expertise and expert performance* (S. 167–184).

Clark, B. & Sharif, B. (2017). ITraceVis: Visualizing Eye Movement Data Within Eclipse. In *Proceedings - 2017 ieee working conference on software visualization, vissoft 2017* (S. 22–32). IEEE. doi: 10.1109/VISSOFT.2017.30

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Aufl.). New York, USA: Lawrence Erlbaum Associates.

Cohen, J. (1992, jun). Statistical Power Analysis. *Current Directions in Psychological Science*, *1* (3), 98–101. Zugriff auf http://journals.sagepub.com/doi/10.1111/1467-8721.ep10768783 doi: 10.1111/1467-8721.ep10768783

Computer Hope. (2017). *What is a stack overflow?* Zugriff am 2021-02-23 auf https://www.computerhope.com/jargon/s/stacover.htm

Cornelissen, F. W., Peters, E. M. & Palmer, J. (2002). The Eyelink Toolbox: Eye tracking with MATLAB and the Psychophysics Toolbox. *Behavior Research Methods, Instruments, Computers*, *34* (4), 613–617. Zugriff auf http://link.springer.com/10.3758/BF03195489https://link.springer.com/content/pdf/10.3758%2FBF03195489.pdf doi: 10.3758/BF03195489

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences* (4), 87–185.

Crosby, M. E., Scholtz, J. & Wiedenbeck, S. (2002). The Roles Beacons Play in Comprehension for Novice and Expert Programmers. In *Proceedings of the 14th workshop of the psychology of programming interest group* (S. 58–73). London, Vereinigtes Königreich. Zugriff auf https://svn.imp.fu-berlin.de/agddi-literatur/ag_lit/Crosbyetal.-2002-TheRolesBeaconsPlayinComprehensionforNovice.pdf

Crosby, M. E. & Stelovsky, J. (1990). How do we read algorithms? A case study. *Computer*, *23* (1), 25–35. Zugriff auf http://ieeexplore.ieee.org/document/48797/ doi: 10.1109/2.48797

Czerwonka, J., Greiler, M. & Tilford, J. (2015). Code Reviews Do Not Find Bugs. How the Current Code Review Best Practice Slows Us Down. In *Proceedings of the 37th ieee/acm international conference on software engineering (icse)* (Bd. 2, S. 27–28). Florenz, Italien: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/7202946/ doi: 10.1109/ICSE.2015.131

Davis, A. M., Bersoff, E. H. & Comer, E. R. (1988). Software System Engineering Process Models. *Software Requirements Engineering*, *14* (10), 1453–1461. Zugriff auf http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=5989269 doi: 10.1109/97811181566674.ch6

Delabarre, E. B. (1898). A Method of Recording Eye-Movements. *The American Journal of Psychology*, *9* (4), 572–574.

Deutsche Gesetzliche Unvallversicherung. (2024). *Vision Zero.* Zugriff am 2024-02-19 auf https://www.dguv.de/de/praevention/visionzero/index.jsp

Diagnostic performance on briefly presented digital pathology images. (2015). *Journal of Pathology Informatics*, *6* (1), 56. doi: 10.4103/2153-3539.168517

Dieterich, E.-W. (2009). *C++*. Oldenbourg Wissenschaftsverlag. Zugriff auf https://www.degruyter.com/view/title/315433 doi: 10.1524/9783486593228

Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, *15* (10), 859–866. Zugriff auf http://portal.acm.org/citation.cfm?doid=355604.361591 doi: 10.1145/355604.361591

Dikmen, M. & Burns, C. (2017). Trust in Autonomous Vehicles. In *Proceedings of the ieee international conference on syystems, man, and cybernetics (smc)* (S. 1093–1098). doi: https://doi.org/10.1109/SMC.2017.8122757

Dodge, R. & Cline, T. S. (1901). The angle velocity of eye movements. *Psychological Review*, *8* (2), 145–157. doi: 10.1037/h0076100

Dong, Y. & Yang, F. (2019). *C++ Programming* (L. Zheng, Hrsg.). Hoboken, NJ, USA: De Gruyter. Zugriff auf https://www.degruyter.com/document/doi/10.1515/9783110471977/html doi: 10.1515/9783110471977

Donovan, T. & Litchfield, D. (2013). Looking for Cancer: Expertise Related Differences in Searching and Decision Making. *Applied Cognitive Psychology*, *27* (1), 43–49. doi: 10.1002/acp.2869

Döring, N. & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften*. Berlin, Heidelberg: Springer. Zugriff auf http://www.lehrbuch-psychologie.dehttp://link.springer.com/10.1007/978-3-642-41089-5 doi: 10.1007/978-3-642-41089-5

Dowson, M. (1997). The Ariane 5 software failure. *ACM SIGSOFT Software Engineering Notes*, *22* (2), 84. Zugriff auf http://portal.acm.org/citation.cfm?doid=251880.251992 doi: 10.1145/251880.251992

Drew, T., Evans, K., Võ, M. L., Jacobson, F. L. & Wolfe, J. M. (2013). Informatics in radiology: What can you see in a single glance and how might this guide visual search in medical images? *Radiographics*, *33* (1), 263–274. doi: 10.1148/rg.331125023

Duchowski, A. T. (2017). *Eye Tracking Methodology* (3rd Aufl.). Cham, Schweiz: Springer. Zugriff auf http://link.springer.com/10.1007/978-3-319-57883-5 doi: 10.1007/978-3-319-57883-5

Duden. (2020). *Expertise.* Zugriff am 2023-09-08 auf https://www.duden.de/rechtschreibung/Expertise

Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A. & Wagner, D. (2013). An empirical study on the effectiveness of security code review. In *Proceedings of the international symposium on engineering secure software and systems (essos)* (S. 197–212). Paris, Frankreich: Springer AG.

Embedded Software Engineering. (2018). *Raus aus der Software-Krise : 50 Jahre Software-Engineering.* Zugriff am 2023-02-21 auf https://www.embedded-software-engineering.de/raus-aus-der-software-krise-50-jahre-software-engineering-a-652bae88f2d67e0ee1be521ce878c14a/

Ericsson, K. A. (2006a). The Influence of Experience and Deliberate Practice on the Development of Superior Expert Performance. In *The cambridge handbook of*

expertise and expert performance (S. 683–704). doi: 10.1017/cbo9780511816796.038

Ericsson, K. A. (2006b). An Introduction to The Cambridge Handbook of Expertise and Expert Performance: Its Development, Organization, and Content. In K. A. Ericsson, N. Charness, P. J. Feltovich & R. R. Hoffman (Hrsg.), *The cambridge handbook of expertise and expert performance* (S. 3–20). Cambridge, Vereinigtes Königreich: Cambridge University Press. Zugriff auf https://www.cambridge.org/core/product/identifier/CBO9780511816796A011/type/book_part doi: 10.1017/CBO9780511816796.001

Ericsson, K. A. (2016). Expertise and individual differences: The search for the structure and acquisition of experts' superior performance. *Wiley Interdisciplinary Reviews: Cognitive Science*, *8* (1), 1–6. doi: 10.1002/wcs.1382

Ericsson, K. A. (2018). The Differential Influence of Experience, Practice, and Deliberate Practice on the Development of Superior Individual Performance of Experts. In *The cambridge handbook of expertise and expert performance* (S. 745–769). Cambridge University Press. Zugriff auf https://www.cambridge.org/core/product/identifier/9781316480748%23CN-bp-38/type/book_part doi: 10.1017/9781316480748.038

Ericsson, K. A., Hoffman, R. R., Kozbelt, A. & Williams, A. M. (Hrsg.). (2018). *The Cambridge Handbook of Expertise and Expert Performance*. Cambridge, Vereinigtes Königreich: Cambridge University Press. Zugriff auf https://www.cambridge.org/core/product/identifier/9781316480748/type/book doi: 10.1017/9781316480748

Ericsson, K. A. & Kintsch, W. (1995). Long-term working memory. *Psychological Review*, *102* (2), 211–245. Zugriff auf http://www.ncbi.nlm.nih.gov/pubmed/3747476http://doi.apa.org/getdoi.cfm?doi=10.1037/0033-295X.102.2.211 doi: 10.1037/0033-295X.102.2.211

Ericsson, K. A., Krampe, R. T. & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, *100* (3), 363–406. Zugriff auf https://graphics8.nytimes.com/images/blogs/freakonomics/pdf/DeliberatePractice(PsychologicalReview).pdf

Ericsson, K. A. & Lehmann, A. C. (1996). Expert and exceptional performance: Evidence of maximal adaptation to task. *Annual Review of Psychology*, *47*, 273–305.

Ericsson, K. A. & Towne, T. J. (2010). Expertise. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1* (3), 404–416. doi: 10.1002/wcs.47

Ernst, H., Schmidt, J. & Beneken, G. (2016). *Grundkurs Informatik* (6th Aufl.). Wiesbaden, Deutschland: Springer. Zugriff auf http://link.springer.com/10.1007/978-3-658-14634-4 doi: 10.1007/978-3-658-14634-4

European Space Agency. (1996). *Ariane 501 explosion.* Zugriff auf

https://www.esa.int/ESA_Multimedia/Images/1998/01/Ariane_501_explosion#.Y_R1v7vNUQY.link

Evans, K. K., Haygood, T. M., Cooper, J., Culpan, A. M. & Wolfe, J. M. (2016). A half-second glimpse often lets radiologists identify breast cancer cases even when viewing the mammogram of the opposite breast. *Proceedings of the National Academy of Sciences of the United States of America*, *113* (37), 10292–10297. doi: 10.1073/pnas.1606187113

Eye movements and vision. (1967). *Neuropsychologia*, *6* (4), 389–390. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/0028393268900122 doi: 10.1016/0028-3932(68)90012-2

Ezer, T., Greiner, M., Grabinger, L., Hauser, F. & Mottok, J. (2023). Eye Tracking As Technology in Education: Data Quality Analysis and Improvements. In *16th annual international conference of education, research and innovation* (S. 4500–4509). Sevilla, Spanien: IATED Academy. doi: 10.21125/iceri.2023.1127

Feltovich, P. J., Prietula, M. J. & Ericsson, K. A. (2006). Studies of Expertise from Psychological Perspectives. In *The cambridge handbook of expertise and expert performance* (S. 41–68). Cambridge University Press. Zugriff auf https://www.cambridge.org/core/product/identifier/CBO9780511816796A015/type/book_part doi: 10.1017/CBO9780511816796.004

Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R*. Los Angeles: SAGE Publications Ltd.

Fitts, P. M., Jones, R. & J.L., M. (1950). Eye movements of aircraft pilots during instrument-landing approaches. *Aeronautical Engineering Review*.

Fiutak, M. (2006). *Ëuropaweit 150 Milliarden Euro Schaden durch fehlerhafte Software* . Zugriff am 2020-07-21 auf http://www.pressetext.com/print/20060126030

Fleetwood, J. (2017). Public health, ethics, and autonomous vehicles. *American Journal of Public Health*, *107* (4), 532–537. doi: 10.2105/AJPH.2016.303628

Förster, A. (2023). *Bis 2050 keine Verkehrstoten mehr in der EU.* Zugriff am 2024-02-19 auf https://www.euranetplus.de/2023/11/30/bis-2050-keine-vverkehrstoten-mehr-in-der-eu/#:~:text=Die%20EU%20hat%20sich%20zum,jetzt%20neuen%20EU-Vorschriften%20zugestimmt

Fox, S. E. & Faulkner-Jones, B. E. (2017). Eye-Tracking in the Study of Visual Expertise: Methodology and Approaches in Medicine. *Frontline Learning Research*, *5* (3), 29–40. Zugriff auf https://journals.sfu.ca/flr/index.php/journal/article/view/258/352 doi: 2295-3159

Friedman, M. (1937, dec). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, *32* (200), 675–701. Zugriff auf http://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522 doi: 10.1080/01621459.1937.10503522

Friedman, M. (1939, mar). A correction: The use of ranks to avoid the as-

sumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *34* (205), 109–109. Zugriff auf http://www.tandfonline.com/doi/abs/10.1080/01621459.1939.10502372 doi: 10.1080/01621459.1939.10502372

Fritz, T., Begel, A., Müller, S. C., Yigit-Elliott, S. & Züger, M. (2014). Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th international conference on software engineering - icse 2014* (S. 402–413). Hyderabad, Indien: ACM Press. Zugriff auf http://doi.acm.org/10.1145/2568225.2568266http://dl.acm.org/citation.cfm?doid=2568225.2568266 doi: 10.1145/2568225.2568266

Geeks for Geeks. (2017). *Object Slicing in C ++.* Zugriff auf https://www.geeksforgeeks.org/object-slicing-in-c/

Geeks for Geeks. (2022). *Object Slicing in C ++.* Zugriff am 2022-06-30 auf https://www.geeksforgeeks.org/object-slicing-in-c/

Geeks for Geeks. (2024). *Copy Constructor in C++.* Zugriff am 2024-02-16 auf https://www.geeksforgeeks.org/copy-constructor-in-cpp/

GeeksforGeeks. (2021). *Virtual Destructor.* Zugriff am 2022-06-27 auf https://www.geeksforgeeks.org/virtual-destructor/

Gegenfurtner, A. (2020). *Professional Vision and Visual Expertise* (Habilitation).

Gegenfurtner, A., Kok, E., van Geel, K., de Bruin, A., Jarodzka, H., Szulewski, A. & van Merriënboer, J. J. (2017). The challenges of studying visual expertise in medical image diagnosis. *Medical Education*, *51* (1), 97–104. doi: 10.1111/medu.13205

Gegenfurtner, A., Lehtinen, E. & Säljö, R. (2011). Expertise Differences in the Comprehension of Visualizations: A Meta-Analysis of Eye-Tracking Research in Professional Domains. *Educational Psychology Review*, *23* (4), 523–552. doi: 10.1007/s10648-011-9174-7

Gegenfurtner, A. & Merriënboer, J. J. G. V. (2017). Methodologies for Studying Visual Expertise. *Frontline Learning Research*, *5* (3), 1–13. doi: 10.14786/flr.v5i3.316

Gerrig, R. J. (2018). Visuelle Wahrnehmung. In T. Dörfler & J. Roos (Hrsg.), *Psychologie* (21., aktua Aufl., S. 135–144). Halbergmoos, Deutschland: Pearson.

Gleick, J. (1996). *A bug and a crash. Sometimes a bug is more than a nuisance.* Zugriff am 2020-07-21 auf https://around.com/ariane.html

Grabinger, L., Hauser, F. & Mottok, J. (2023a). Evaluating graph-based modeling languages. In *Proceedings of the 5th european conference on software engineering education* (S. 120–129). Seeon, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3593663.3593664 doi: 10.1145/3593663.3593664

Grabinger, L., Hauser, F. & Mottok, J. (2023b). On the perception of graph layouts. *Journal of Software: Evolution and Process*. Zugriff auf https://onlinelibrary.wiley.com/doi/10.1002/smr.2599 doi: 10.1002/smr.2599

Greive, M. & Stiens, T. (2021). *So könnte ein deutsches Digitalministerium aussehen.* Zugriff am 2021-10-27 auf https://www.handelsblatt.com/politik/deutschland/bundespolitik-so-koennte-ein-deutsches-digitalministerium-aussehen/27639506.html?ticket=ST-1007986-bZccBMUzmLjuyQlGXV9S-cas01.example.org

Grotelüschen, F. (2008). *Der Absturz der Ariane 5.* Zugriff am 2020-07-21 auf https://www.deutschlandfunk.de/der-absturz-der-ariane-5.676.de.html?dram:article{_}id=25637

Gruber, H. (2007). Bedingungen von Expertise. *Begabt sein in Deutschland* (25), 93–112. Zugriff auf http://books.google.com/books?hl=en&lr=&id=CpmZTK3CGwMC&oi=fnd&pg=PA93&dq=Bedingungen+von+Expertise&ots=HLWbMqJXHa&sig=KruTvuA6iIaFs9uKbSF_hXiRfGM

Gruber, H., Harteis, C. & Rehrl, M. (2004). Wissensmanagement und Expertise (Forschungsbericht Nr.9). *Der Mensch im Wissensmanagement: Psychologische Konzepte zum besseren Verständnis und Umgang mit Wissen* (9), 79–88.

Gruber, H., Jansen, P., Marienhagen, J. & Altenmüller, E. (2010). Adaptations During the Acquisition of Expertise. *Talent Development Excellence*, *2* (1), 3–15.

Gruber, H. & Lehmann, A. C. (2008). Entwicklung von Expertise und Hochleistung in Musik und Sport. *Angewandte Entwicklungspsychologie*, *7* (26), 497–520.

Gruber, H., Lehtinen, E., Palonen, T. & Degner, S. (2008). Persons in the shadow: Assessing the social context of high abilities. *Psychology Science Quarterly*, *50*, 237–258.

Guarnera, D. T., Bryant, C. A., Mishra, A., Maletic, J. I. & Sharif, B. (2018). iTrace: Eye Tracking Infrastructure for Development Environments. In *Proceedings of the 2018 acm symposium on eye tracking research & applications - etra '18* (S. 1–3). New York, New York, USA: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3204493.3208343 doi: 10.1145/3204493.3208343

Hacker, W. (1992). *Expertenkönnen, Erkennen und Vermitteln*. Göttingen, Deutschland: Hogrefe.

Haider, H. & Frensch, P. A. (1996). The role of information reduction in skill acquisition. *Cognitive Psychology*, *30* (3), 304–337. doi: 10.1006/cogp.1996.0009

Haider, H. & Frensch, P. A. (1999). Information reduction during skill acquisition: The influence of task instruction. *Journal of Experimental Psychology: Applied*, *5* (2), 129–151. doi: 10.1037/1076-898X.5.2.129

Hart, S. G. & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. , *52*, 139–183. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0166411508623869 doi: 10.1016/S0166-4115(08)62386-9

Haselhuhn, T. (2020). *Zur Erfassung kognitiver Lernstrategien mit Blickbewegungen. Experimentelle Untersuchungen mit Studierenden beim Textlernen* (Doktorarbeit, Universität Regensburg). Zugriff auf http://www.elsevier.com/locate/scp

Hauser, F., Grabinger, L., Mottok, J. & Gruber, H. (2023). Visual expertise in co-

de reviews. Using holistic models of image perception to analyze and interpret eye movements. In *Proceedings of the symposium on eye tracking research and applications (etra)* (S. 1–7). Tübingen, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3588015.3589189 doi: 10.1145/3588015.3589189

Hauser, F., Grabinger, L., Mottok, J., Jahn, S. & Nadimpalli, V. K. (2023). The expert's view: Eye movement modeling examples in software engineering education. In *Proceedings of the 5th european conference on software engineering education* (S. 148–152). Seeon, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3593663.3593683 doi: 10.1145/3593663.3593683

Hauser, F., Mottok, J. & Gruber, H. (2018). Eye tracking metrics in software engineering. In *Proceedings of the 3rd european conference on software engineering education* (S. 39–44). Seeon, Deutschland. doi: 10.1145/3209087.3209092

Hauser, F., Reiß, M., Nivala, M., Mottok, J. & Gruber, H. (2017). Eye Tracking Applied: Visual Expertise in Code Reviews. In *Proceedings of the international conference on education and new learning technologies (edulearn)* (Bd. 1, S. 379–389). Barcelona, Spanien: IATED Academy. doi: 10.21125/edulearn.2017.1084

Hauser, F., Reuter, R., Gegenfurtner, A., Gruber, H. G. & Mottok, J. (2019). Eye movements in software modelling - What do they tell us about heuristics? In *Proceedings of the international conference of education, research and innovation (iceri)* (S. 6064–6070). Sevilla, Spanien: IATED Academy. Zugriff auf http://library.iated.org/view/HAUSER2019EYE doi: 10.21125/iceri.2019.1469

Hauser, F., Reuter, R., Hutzler, I., Mottok, J. & Gruber, H. (2018). Eye Movements in Software Engineering - What Differs the Expert From the Novice? In *Proceedings of the international conference of education, research and innovation (iceri)* (S. 632–642). Sevilla, Spanien: IATED Academy. doi: 10.21125/iceri.2018.1129

Hauser, F., Schreistetter, S., Reuter, R., Mottok, J. H., Gruber, H., Holmqvist, K. & Schorr, N. (2020). Code Reviews in C++. In *Proceedings of the symposium on eye tracking research and applications (etra)* (S. 1–5). Stuttgart, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3379156.3391980 doi: 10.1145/3379156.3391980

Hauser, F., Stark, T., Mottok, J., Gruber, H. & Reuter, R. (2020). Deliberate Practice in Programming. In *Proceedings of the 4th european conference on software engineering education* (S. 42–46). Seeon, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3396802.3396815 doi: 10.1145/3396802.3396815

Hejmady, P. & Narayanan, N. H. (2012). Visual attention patterns during program debugging with an IDE. In *Eye tracking research applications (etra)* (Bd. 1, S. 197–200). Santa Barbara, CA, USA: ACM. doi: 10.1145/2168556.2168592

Hevelke, A. & Nida-Rümelin, J. (2015). Responsibility for Crashes of Autonomous Vehicles: An Ethical Analysis. *Science and Engineering Ethics*, *21* (3), 619–630. doi: 10.1007/s11948-014-9565-5

Hmelo-Silver, C. (2004, feb). Comparing expert and novice understanding of a com-

plex system from the perspective of structures, behaviors, and functions. *Cognitive Science*, *28* (1), 127–138. Zugriff auf http://doi.wiley.com/10.1016/S0364-0213(03)00065-X doi: 10.1016/S0364-0213(03)00065-X

Hoffmann, D. W. (2013). *Software-Qualität*. Berlin: Springer.

Holding, D. H. (1985). *The psychology of chess skill*. Hilldale, USA: Erlbaum.

Holmqvist, K. & Andersson, R. (2017). *Eye tracking: A comprehensive guide to methods, paradigms, and measures* (2nd Aufl.). Charlston, SC, USA: CreateSpace Independent Publishing Platform.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H. & Van De Weijer, J. (2011). *Eye tracking: A comprehensive guide to methods and measures*. Oxford, Vereinigtes Königreich: Oxford University Press.

Holyoak, K. J. (1991). Symbolic connectionism: Toward third-generation theories of expertise. In K. A. Ericsson & J. Smith (Hrsg.), *Toward a general theory of expertise: Prospects and limits* (S. 301–335). Cambridge, Vereinigtes Königreich: Cambridge University Press.

Homann, A., Grabinger, L., Hauser, F. & Mottok, J. (2023). An Eye Tracking Study on MISRA C Coding Guidelines. In *Proceedings of the 5th european conference on software engineering education* (S. 130–137). Seeon, Deutschland: ACM. Zugriff auf https://dl.acm.org/doi/10.1145/3593663.3593671 doi: 10.1145/3593663.3593671

Hou, T.-Y., Lin, Y.-T., Lin, Y.-C., Chang, C.-H. & Yen, M.-H. (2013). Exploring the Gender Effect on Cognitive Processes in Program Debugging based on Eye-movement Analysis. In *Proceedings of the 5th international conference on computer supported education* (S. 469–473). Aachen: SciTePress - Science and and Technology Publications. Zugriff auf http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004415104690473 doi: 10.5220/0004415104690473

Hryszko, J. & Madeyski, L. (2017). Assessment of the Software Defect Prediction Cost Effectiveness in an Industrial Project. In L. Madeyski, M. Śmiałek, B. Hnatkowska & Z. Huzar (Hrsg.), *Software engineering: Challenges and solutions* (S. 77–90). Worclaw, Polen: Springer. Zugriff auf http://link.springer.com/10.1007/978-3-319-43606-7{_}6 doi: 10.1007/978-3-319-43606-7_6

Huey, E. B. (1898). Preliminary experiments in the physiology and psychology of reading. *The American Journal of Psychology*, *9* (4), 575–586. Zugriff auf http://www.jstor.org/stab

Hutzler, I., Hauser, F., Reuter, R., Mottok, J. & Gruber, H. (2018). Will the Noun/Verb Analysis Be Used To Generate Class Diagrams? an Eye Tracking Study. In *Proceedings of the international conference of education, research and innovation (iceri)* (S. 505–514). Sevilla, Spanien: IATED Academy. Zugriff auf http://library.iated.org/view/HUTZLER2018WIL doi: 10.21125/iceri.2018.1103

IMotions. (2022). *SMI 250 RED mobile.* Zugriff auf https://imotions.com/hardware/smi-red250mobile/

Indriasari, T. D., Luxton-Reilly, A. & Denny, P. (2020). A Review of Peer Code Review in Higher Education. *ACM Transactions on Computing Education*, *20* (3). doi: 10.1145/3403935

Intel Corporation. (2010). *Base class has non-virtual destructor.* Zugriff am 2022-06-27 auf https://www.smcm.iqfr.csic.es/docs/intel/ssadiag_docs/pt_reference/references/sc_cpp_non_virtual_public_dtor.htm

ISO C++. (2022). *Inheritance — virtual functions.* Zugriff am 2022-06-30 auf https://isocpp.org/wiki/faq/virtual-functions doi: https://isocpp.org/wiki/faq/virtual-functions

Issenberg, S. B., McGaghie, W. C., Petrusa, E. R., Gordon, D. L. & Scalese, R. J. (2005). Features and uses of high-fidelity medical simulations that lead to effective learning: a BEME systematic review. *Medical Teacher*, *27* (1), 10–28. Zugriff auf http://www.tandfonline.com/doi/full/10.1080/01421590500046924 doi: 10.1080/01421590500046924

Jarodzka, H., Scheiter, K., Gerjets, P. & van Gog, T. (2010). In the eyes of the beholder: How experts and novices interpret dynamic stimuli. *Learning and Instruction*, *20* (2), 146–154. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0959475209000255 doi: 10.1016/j.learninstruc.2009.02.019

Jbara, A. & Feitelson, D. G. (2015). How Programmers Read Regular Code: A Controlled Experiment Using Eye Tracking. In *2015 ieee 23rd international conference on program comprehension* (Bd. 22, S. 244–254). Florenz, Italien: IEEE. Zugriff auf http://link.springer.com/10.1007/s10664-016-9477-xhttp://ieeexplore.ieee.org/document/7181453/ doi: 10.1109/ICPC.2015.35

Johnston, P. & Harris, R. (2019). The Boeing 737 MAX Saga: Lessons for Software Organizations. *Software Quality Professional*, *21* (3), 5–12. Zugriff auf www.asq.org

Józsa, E. & Hámornik, B. P. (2012). Find The Difference! Eye Tracking Study on Information Seeking Behavior Using an Online Game. *Journal of Eye tracking Visual Cognition and Emotion*, *2* (1), 27–35. Zugriff auf http://revistas.ulusofona.pt/index.php/JETVCE/article/view/2718

Kalra, N. & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, *94*, 182–193. doi: 10.1016/j.tra.2016.09.010

Kanojia, D. (2020). *Saccades amplitude or saccade length calculation for each saccade in degree?* Zugriff am 2023-12-13 auf https://www.researchgate.net/post/Saccades-amplitude-or-saccade-length-calculation-for-each-saccade-in-degree

Keil, S. (2016). Understanding Junior Doctors' Learning and Performance in Clinical Practice. Zugriff auf https://epub.uni-regensburg.de/34469/ doi:

10.1016/j.crme.2007.07.001

Kevic, K. (2017). Using eye gaze data to recognize task-relevant source code better and more fine-grained. In *Proceedings - 2017 ieee/acm 39th international conference on software engineering companion, icse-c 2017* (S. 103–105). Buenos Aires, Argentinien: IEEE. doi: 10.1109/ICSE-C.2017.152

Kevic, K., Walters, B. M., Shaffer, T. R., Sharif, B., Shepherd, D. C. & Fritz, T. (2015). Tracing software developers' eyes and interactions for change tasks. In *Proceedings of the joint meeting on foundations of software engineering (esec/fse)* (S. 202–213). Bergamo, Italien: ACM. Zugriff auf http://dl.acm.org/citation.cfm?doid=2786805.2786864 doi: 10.1145/2786805.2786864

Kintsch, W., Patel, V. L. & Ericsson, K. A. (1999). The role of long-term working memory in text comprehension. *Psychologia*, *42* (4), 186–198.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J. & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology*, *51* (1), 7–15. Zugriff auf http://dx.doi.org/10.1016/j.infsof.2008.09.009 doi: 10.1016/j.infsof.2008.09.009

Kneuper, R. (2014). Vorgehensmodelle und Reifegradmodelle - Ergänzung oder Konkurrenz? In *Konferenzband der 37. jahrestagung der gesellschaft für informatik e.v. (gi).* Bremen, Deutschland.

Knobloch, L. (2019, oct). *Uni Regensburg baut Informatik aus.* Regensburg. Zugriff am 2020-02-13 auf https://www.mittelbayerische.de/bayern-nachrichten/uni-regensburg-baut-informatik-aus-21705-art1834026.html

Kok, E. M. (2016). *Developing visual expertise. From Shades of Grey to Diagnostic Reasoning in Radiology* (PhD-Thesis). University of Maastricht, Masstricht.

Kok, E. M., de Bruin, A. B., Robben, S. G. & van Merriënboer, J. J. (2012). Looking in the same manner but seeing it differently: Bottom-up and expertise effects in radiology. *Applied Cognitive Psychology*, *26* (6), 854–862. doi: 10.1002/acp.2886

Kok, E. M., Jarodzka, H., de Bruin, A. B., BinAmir, H. A., Robben, S. G. & van Merriënboer, J. J. (2016). Systematic viewing in radiology: Seeing more, missing less? *Advances in Health Sciences Education*, *21* (1), 189–205. doi: 10.1007/s10459-015-9624-y

Kononenko, O., Baysal, O. & Godfrey, M. W. (2016). Code review quality. In *Proceedings of the 38th ieee/acm international conference on software engineering (icse)* (S. 1028–1038). Austin, Texas, USA: IEEE. Zugriff auf http://dl.acm.org/citation.cfm?doid=2884781.2884840 doi: 10.1145/2884781.2884840

Konopka, M., Talian, A., Tvarozek, J. & Navrat, P. (2018). Data flow metrics in program comprehension tasks. In *Proceedings of the workshop on eye movements in programming - emip '18* (S. 1–6). Warschau, Polen: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3216723.3216728 doi: 10.1145/3216723.3216728

Krems, J. (1996). Expertise und Flexibilität. In H. Gruber & A. Ziegler (Hrsg.), *Expertiseforschung. theoretische und methodische grundlagen* (S. 80–91). Opladen, Deutschland: Westdeutscher Verlag.

Krupinski, E. A., Graham, A. R. & Weinstein, R. S. (2013). Characterizing the development of visual search expertise in pathology residents viewing whole slide images. *Human Pathology*, *44* (3), 357–364. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S0046817712002006 doi: 10.1016/j.humpath.2012.05.024

Kundel, H. L. & La Follette, P. S. (1972). Visual Search Patterns and Experience with Radiological Images. *Radiology*, *103* (3), 523–528. Zugriff auf http://pubs.rsna.org/doi/10.1148/103.3.523 doi: 10.1148/103.3.523

Kundel, H. L., Nodine, C. F., Conant, E. F. & Weinstein, S. P. (2007). Holistic Component of Image Perception in Mammogram Interpretation: Gaze-tracking Study. *Radiology*, *242* (2), 396–402. Zugriff auf http://pubs.rsna.org/doi/10.1148/radiol.2422051997 doi: 10.1148/radiol.2422051997

Lamnek, S. & Krell, C. (2016). *Qualitative Sozialforschung* (6. Aufl. Aufl.). Weinheim, Deutschland: Beltz.

Langer, A. M. (2016). *Guide to Software Development*. Cham: Springer. doi: 10.1007/978-1-4471-6799-0

Learn C++. (2022). *Pointers and references to the base class of derived objects.* Zugriff am 2022-06-30 auf https://www.learncpp.com/cpp-tutorial/pointers-and-references-to-the-base-class-of-derived-objects/

LearnCpp.com. (2020). *Object slicing.* Zugriff auf https://www.learncpp.com/cpp-tutorial/object-slicing/

Lee, R. (2019). *Software Engineering Research, Management and Applications*. Cham: Springer.

Lemahieu, W. & Wyns, B. (2010). Low cost eye tracking for human-machine interfacing. *Journal of Eye Tracking, Visual Cognition and Emotion*, *1* (1), 1–12. Zugriff auf http://recil.grupolusofona.pt/bitstream/handle/10437/2297/Lowcosteyetracking.pdf?sequence=1

Leschnik, A. (2020). *Visuelle Wahrnehmung.* Wiesbaden, Deutschland: Springer Fachmedien Wiesbaden. Zugriff auf http://link.springer.com/10.1007/978-3-658-30877-3 doi: 10.1007/978-3-658-30877-3

Lettnin, D. & Winterholer, M. (2017). *Embedded Software Verification and Debugging* (D. Lettnin & M. Winterholer, Hrsg.). New York, NY: Springer New York. Zugriff auf https://books.google.co.in/books?id={_}9GyDgAAQBAJ{&}pg=PA15{&}lpg=PA15{&}dq=N.Dershowitz,Softwarehorrorstories.{%}5BOnline{%}5D.Available:http://www.+cs.tau.ac.il/+nachumd/verify/horror.html{&}source=bl{&}ots=JSSmXqEPdQ{&}sig=7qfhm3J8ggChKWSHPz0r-vuXS6Q{&}hl=en{&}sa=X{&}ved=0ahhttp://link.springer.com/10.1007/978-1-4614-2266-2 doi: 10.1007/978-1-4614-2266-2

Leveson, N. G. (2004). Role of software in spacecraft accidents. *Journal of Spacecraft and Rockets*, *41* (4), 564–575. doi: 10.2514/1.11950

Liang, C., Karolus, J., Kosch, T. & Schmidt, A. (2018). On the Suitability of Real-Time Assessment of Programming Proficiency using Gaze Properties. In *Proceedings of the 7th acm international symposium on pervasive displays - perdis '18* (S. 1–2). München, Deutschland: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3205873.3210702 doi: 10.1145/3205873.3210702

Lin, Y.-T., Wu, C.-C., Hou, T.-Y., Lin, Y.-C., Yang, F.-Y. & Chang, C.-H. (2016, aug). Tracking Students' Cognitive Processes During Program Debugging—An Eye-Movement Approach. *IEEE Transactions on Education*, *59* (3), 175–186. Zugriff auf http://ieeexplore.ieee.org/document/7312518/ doi: 10.1109/TE.2015.2487341

Lowe, R. K. (1999). Extracting information from an animation during complex visual learning. *European Journal of Psychology of Education*, *14* (2), 225–244.

Ludewig, J. & Lichter, H. (2010a). Software-Qualität. In *Software engineering* (S. 65–70).

Ludewig, J. & Lichter, H. (2010b). Software-Qualitätssicherung und -Prüfung. In *Software engineering* (S. 269–294). Heidelberg: dpunkt.

Ludewig, J. & Lichter, H. (2013). *Software Engineering Grundlagen, Menschen, Prozesse, Techniken* (3rd Aufl.). Heidelberg: dpunkt.

Madesh, S. (2015). *Software Development Life Cycle(SDLC) Phases.* Zugriff am 2020-08-12 auf http://theblogreaders.com/software-development-life-cyclesdlc-phases/

Mandl, H. & Spada, H. (1988). *Wissenspsychologie*. München, Deutschland: PVU.

Mangaroska, K., Sharma, K., Giannakos, M., Trætteberg, H. & Dillenbourg, P. (2018, jun). Gaze insights into debugging behavior using learner-centred analysis. In *Proceedings of the 8th international conference on learning analytics and knowledge - lak '18* (Bd. 113, S. 350–359). Sydney, Australien: ACM Press. Zugriff auf http://aip.scitation.org/doi/10.1063/1.4809761http://dl.acm.org/citation.cfm?doid=3170358.3170386 doi: 10.1145/3170358.3170386

Mäntylä, M. V. & Lassenius, C. (2009, may). What Types of Defects Are Really Discovered in Code Reviews? *IEEE Transactions on Software Engineering*, *35* (3), 430–448. Zugriff auf http://ieeexplore.ieee.org/document/4604671/ doi: 10.1109/TSE.2008.71

Masood, I. (2020). *Software Development Life Cycle (SDLC): The Guide.* Zugriff am 2020-08-12 auf https://blog.tara.ai/software-development-life-cycle/

Maxim, B. R. & Kessentini, M. (2016). An introduction to modern software quality assurance. In I. Mistrik, N. Ali, B. Tekinerdogan, R. Soley & J. Grundy (Hrsg.), *Software quality assurance* (S. 19–46). Amsterdam, Niederlande: Elsevier. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/

B9780128023013000028 doi: 10.1016/B978-0-12-802301-3.00002-8

Mayring, P. (2016). *Einführung in die qualitative Sozialforschung*. Weinheim, Deutschland: Beltz.

Melo, J., Narcizo, F. B., Hansen, D. W., Brabrand, C. & Wasowski, A. (2017). Variability through the Eyes of the Programmer. In *Ieee international conference on program comprehension* (S. 34–44). Buenos Aires, Argentinien: IEEE. doi: 10.1109/ICPC.2017.34

Merriam-Webster. (2020). *Expertise.* Zugriff am 2020-02-20 auf https://www.merriam-webster.com/dictionary/expertise

Microsoft. (2009). *Expertise.* Zugriff am 2017-02-14 auf http://encarta.msn.com/dictionary/expertise.html

Moore, G. E. (1965). Cramming more components onto integrated circuits (Reprinted from Electronics. *Proceedings Of The Ieee*, *38* (8), 114–117. Zugriff auf papers3://publication/uuid/8E5EB7C8-681C-447D-9361-E68D1932997D doi: 10.1109/N-SSC.2006.4785860

Moore, M. M. (2002). Software Engineering Education. *IEEE Software*, *19* (5), 103. doi: 10.1109/MS.2002.1032866

Moser, M. (2022). *Comparing the impact of code highlighting schemes during code comprehension using eye-tracking technologies* (Dissertation, Ostbayerische Technische Hochschule Regensburg). Zugriff auf https://github.com/markusmo3/oth-master-thesis/blob/main/MasterThesis.pdf

Müller, H. & Weichert, F. (2012). *Vorkurs Informatik*. Wiesbaden, Deutschland: Vieweg+Teubner Verlag. Zugriff auf http://link.springer.com/10.1007/978-3-8348-9861-6 doi: 10.1007/978-3-8348-9861-6

Naur, P. & Randell, B. (1968). Software Engineering. In *Report on a conference sponsored by the nato scince committee.* Garmisch, Deutschland: NATO Scientific Affairs Division.

Nivala, M., Hauser, F., Mottok, J. & Gruber, H. (2016). Developing visual expertise in software engineering: An eye tracking study. In *Global engineering education conference (educon '16)* (S. 613–620). Abu Dhabi: IEEE Inc. Zugriff auf http://ieeexplore.ieee.org/document/7474614/ doi: 10.1109/EDUCON.2016.7474614

Nodine, C. F. & Kundel, H. L. (1987). The Cognitive Side of Visual Search in Radiology. In *Eye movements from physiology to cognition* (S. 573–582). Elsevier. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/B9780444701138500813 doi: 10.1016/B978-0-444-70113-8.50081-3

Nodine, C. F. & Kundel, H. L. (1997). Al Hirschfeld's NINA as a prototype search task for studying perceptual error in radiology. In H. L. Kundel (Hrsg.), *Medical imaging* (S. 308–312). Newport Beach, CA, USA. Zugriff auf http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=922081 doi: 10.1117/12.271306

Nodine, C. F., Kundel, H. L., Lauver, S. C. & Toto, L. C. (1996). Nature of expertise

in searching mammograms for breast masses. *Academic Radiology*, *3* (12), 1000–1006. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S1076633296800328 doi: 10.1016/S1076-6332(96)80032-8

Nodine, C. F. & Mello-Thoms, C. (2000). The Nature of Expertise in Radiology. In *Handbook of medical imaging, volume 1. physics and psychophysics* (S. 859–894). Bellingham, WA, USA: SPIE. Zugriff auf http://ebooks.spiedigitallibrary.org/content.aspx?doi=10.1117/3.832716.ch19 doi: 10.1117/3.832716.ch19

Nodine, C. F. & Mello-Thoms, C. (2010). The role of expertise in radiologic image interpretation. In E. Samei & E. A. Krupinski (Hrsg.), *The handbook of medical image perception and techniques* (S. 139–156). Cambridge, Vereinigtes Königreich: Cambridge University Press.

Nodine, C. F., Mello-Thoms, C., Kundel, H. L. & Weinstein, S. P. (2002, oct). Time Course of Perception and Decision Making During Mammographic Interpretation. *American Journal of Roentgenology*, *179* (4), 917–923. Zugriff auf http://www.ajronline.org/doi/10.2214/ajr.179.4.1790917 doi: 10.2214/ajr.179.4.1790917

Nyström, M., Niehorster, D. C., Andersson, R. & Hooge, I. (2021). The Tobii Pro Spectrum: A useful tool for studying microsaccades? *Behavior Research Methods*, *53* (1), 335–353. doi: 10.3758/s13428-020-01430-3

Obaidellah, U., Al Haek, M. & Cheng, P. C.-H. (2018, jan). A Survey on the Usage of Eye-Tracking in Computer Programming. *ACM Computing Surveys*, *51* (1), 1–58. Zugriff auf http://dl.acm.org/citation.cfm?doid=3177787.3145904 doi: 10.1145/3145904

Obaidellah, U., Raschke, M. & Blascheck, T. (2019). Classification of strategies for solving programming problems using AoI sequence analysis. In *Proceedings of the 11th acm symposium on eye tracking research applications - etra '19* (S. 1–9). Denver, CO: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3314111.3319825 doi: 10.1145/3314111.3319825

OpenAI. (2024). *Veröffentlichung von ChatGPT.* Zugriff am 2024-01-29 auf https://chat.openai.com/

Ostbayerische Technische Hochschule Regensburg. (2020a). *Dr. Markus Söder informiert sich über KI-Kompetenzen der OTH Regensburg.* Zugriff am 2020-06-28 auf https://www.oth-regensburg.de/de/weiterbildung/nachrichten/einzelansicht/news/dr-markus-soeder-informiert-sich-ueber-ki-kompetenzen-der-oth-regensburg.html

Ostbayerische Technische Hochschule Regensburg. (2020b). *Neu: Bachelor Künstliche Intelligenz und Data Science.* Zugriff am 2020-06-28 auf https://www.oth-regensburg.de/de/weiterbildung/nachrichten/einzelansicht/news/dr-markus-soeder-informiert-sich-ueber-ki-kompetenzen-der-oth-regensburg.html

Peachock, P., Iovino, N. & Sharif, B. (2017). Investigating Eye Mo-

vements in Natural Language and C++ Source Code - A Replication Experiment. In *Proceedings of the augmented cognition (ac)* (S. 206–218). Vancouver: Springer. Zugriff auf https://www.scopus.com/inward/record.uri?eid=2-s2.0-85025175768{&}doi=10.1007{%}2F978-3-319-58628-1{_}17{&}partnerID=40{&}md5=ea793e7cf3c83adf1d4a6aaeafc4c9d6https://link.springer.com/10.1007/978-3-319-58628-1{_}17 doi: 10.1007/978-3-319-58628-1_17

Pencil Programmer. (2022). *C ++ Override keyword.* Zugriff am 2022-06-30 auf https://pencilprogrammer.com/cpp-tutorials/override-keyword/

Peng, F., Li, C., Song, X., Hu, W. & Feng, G. (2016, jun). An Eye Tracking Research on Debugging Strategies towards Different Types of Bugs. In *2016 ieee 40th annual computer software and applications conference (compsac)* (Bd. 40, S. 130–134). Atlanta, GA: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/7552192/ doi: 10.1109/COMPSAC.2016.57

Peterson, C. S., Abid, N. J., Bryant, C. A., Maletic, J. I. & Sharif, B. (2019). Factors influencing dwell time during source code reading. In *Proceedings of the 11th acm symposium on eye tracking research applications - etra '19* (S. 1–4). Denver, CO: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3314111.3319833 doi: 10.1145/3314111.3319833

Posner, M. I. (1988). What is it to be an expert? In M. T. H. Chi, R. Glaser & M. J. Farr (Hrsg.), *The nature of expertise* (S. 29–36). Hillsdale, USA: Erlbaum.

Rai, L. (2019). *Programming in C++* (L. Rai, Hrsg.). De Gruyter. Zugriff auf https://www.degruyter.com/document/doi/10.1515/9783110593846/html doi: 10.1515/9783110593846

Rashid, J. & Nisar, M. W. (2016). How to Improve a Software Quality Assurance In Software Development-A Survey. *International Journal of Computer Science and Information Security*, *14* (8), 99–108. Zugriff auf https://search.proquest.com/docview/1876023819?accountid=43603

Reingold, E. M. & Sheridan, H. (2012). Eye movements and visual expertise in chess and medicine. *The Oxford Handbook of Eye Movements*, 528–550. doi: 10.1093/oxfordhb/9780199539789.013.0029

Reuter, R., Beslmeisl, M. & Mottok, J. (2017, apr). Work in progress: Teaching-obstacles in higher software engineering education. In *Proceedings of the ieee global engineering education conference (educon)* (Bd. 19, S. 1631–1635). Athen, Griechenland: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/1032866/http://ieeexplore.ieee.org/document/7943067/ doi: 10.1109/EDUCON.2017.7943067

Rezagholi, M. (2004). *Prozess- und Technologiemanagement in der Softwareentwicklung*. München, Deutschland: Oldenburg Wissenschaftsverlag.

Rodeghero, P., Liu, C., McBurney, P. W. & McMillan, C. (2015, nov). An Eye-Tracking Study of Java Programmers and Application to Source Code Summarization. ,

41 (11), 1038–1054. Zugriff auf http://ieeexplore.ieee.org/document/7118751/ doi: 10.1109/TSE.2015.2442238

Rodeghero, P. & McMillan, C. (2015, oct). An Empirical Study on the Patterns of Eye Movement during Summarization Tasks. In *Proceedings of the international symposium on empirical software engineering and measurement (esem)* (S. 11–20). Beijing, China: ACM/IEEE. Zugriff auf http://ieeexplore.ieee.org/document/7321188/ doi: 10.1109/ESEM.2015.7321188

Rodeghero, P., McMillan, C., McBurney, P. W., Bosch, N. & D'Mello, S. (2014). Improving automated source code summarization via an eye-tracking study of programmers. In *Proceedings of the 36th international conference on software engineering* (S. 390–401). Hyderabad, India: ACM. Zugriff auf http://doi.acm.org/10.1145/2568225.2568247 doi: 10.1145/2568225.2568247

Romero, P., Boulay, B., Cox, R. & Lutz, R. (2003). Java debugging strategies in multi-representational environments. In *Psychology of programming languages interest group 15th workshop* (S. 421–435). Staffordshire.

Romero, P., Cox, R., Boulay, B. D. & Lutz, R. (2004). Diagrammatic Representation and Inference. In *Third international conference on theory and application of diagrams* (S. 344–346). Zugriff auf http://link.springer.com/10.1007/b95854 doi: 10.1007/b95854

Rubin, G. D., Roos, J. E., Tall, M., Harrawood, B., Bag, S., Ly, D. L., ... Roy Choudhury, K. (2015). Characterizing Search, Recognition, and Decision in the Detection of Lung Nodules on CT Scans: Elucidation with Eye Tracking. *Radiology*, *274* (1), 276–286. Zugriff auf http://pubs.rsna.org/doi/10.1148/radiol.14132918 doi: 10.1148/radiol.14132918

Saddler, J. A. (2019). Looks can mean achieving: Understanding Eye Gaze Patterns of Proficiency in Code Comprehension. In *Proceedings of the 11th acm symposium on eye tracking research applications - etra '19* (S. 1–3). Denver, CO: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3314111.3322876 doi: 10.1145/3314111.3322876

Sawyer, K. (1999). *Mystery of Orbiter Crash Solved.* Zugriff auf https://www.washingtonpost.com/wp-srv/national/longterm/space/stories/orbiter100199.htm#:~:text=NASA%27s%20Mars%20Climate%20Orbiter%20was,Martian%20surface%2C%20investigators%20said%20yesterday.

Schnappinger, M., Osman, M. H., Pretschner, A., Pizka, M. & Fietzke, A. (2018). Software quality assessment in practice. In *Proceedings of the 12th acm/ieee international symposium on empirical software engineering and measurement (esem)* (S. 1–6). Oulu, Finnland: ACM. Zugriff auf https://doi.org/10.1145/3239235.3268922https://dl.acm.org/doi/10.1145/3239235.3268922 doi: 10.1145/3239235.3268922

Schneider, W. (1993). Acquiring expertise: Determinants of exceptional performance. *International handbook of research and development of giftedness and talent.*, 311–324.

Zugriff auf http://psycnet.apa.org/psycinfo/1993-99039-014

Schorr, N. (o. J.). *Die Rolle von Farbcodierung und Programmierexpertise auf die Fehlersuche im Quellcode Masterarbeit Universität Regensburg* (Masterarbeit). Universität Regensburg.

SensoMotoric Instruments. (2017). *BeGaze Manual*. Teltow.

Shaffer, T. R., Wise, J. L., Walters, B. M., Müller, S. C., Falcone, M. & Sharif, B. (2015). iTrace: Enabling eye tracking on software artifacts within the IDE to support software engineering tasks. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering - esec/fse 2015* (S. 954–957). Bergamo, Italy: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=2786805.2803188 doi: 10.1145/2786805.2803188

Shapiro, F. R. (1987). Etymology of the Computer Bug: History and Folklore. *American Speech*, *62* (4), 376. Zugriff auf https://www.jstor.org/stable/455415?origin=crossref doi: 10.2307/455415

Sharafi, Z., Sharif, B., Guéhéneuc, Y. G., Begel, A., Bednarik, R. & Crosby, M. (2020). A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering*, *25* (5), 3128–3174. doi: 10.1007/s10664-020-09829-4

Sharafi, Z., Soh, Z. & Guéhéneuc, Y.-G. (2015). A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology*, *67* (7), 79–107. Zugriff auf http://dx.doi.org/10.1016/j.infsof.2015.06.008 doi: 10.1016/j.infsof.2015.06.008

Sharafi, Z., Soh, Z., Gueheneuc, Y.-G. & Antoniol, G. (2012). Women and Men – Different but Equal: On the Impact of Identifier Style on Source Code Reading. In *20th ieee international conference on program comprehension (icpc)* (S. 27–36). IEEE. doi: 10.1109/icpc.2012.6240505

Sharif, B., Falcone, M. & Maletic, J. I. (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)* (S. 381). New York, New York: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=2168556.2168642 doi: 10.1145/2168556.2168642

Sharif, B., Jetty, G., Aponte, J. & Parra, E. (2013). An empirical study assessing the effect of SeeIT 3D on comprehension. *2013 1st IEEE Working Conference on Software Visualization - Proceedings of VISSOFT 2013*. doi: 10.1109/VISSOFT.2013.6650519

Sharif, B. & Maletic, J. I. (2010a). An Eye Tracking Study on camelCase and under_score Identifier Styles. In *Proceedings of the 18th ieee international conference on program comprehensioni* (S. 196–205). Braga, Italien: IEEE. Zugriff auf http://ieeexplore.ieee.org/document/5521745/ doi: 10.1109/ICPC.2010.41

Sharif, B. & Maletic, J. I. (2010b). An eye tracking study on the effects of layout in understanding the role of design patterns. *IEEE International Conference on Software Maintenance, ICSM*. doi: 10.1109/ICSM.2010.5609582

Sharif, B. & Mansoor, N. (2022). Humans in Empirical Software Engineering Studies: An Experience Report. In *29th edition of the ieee international conference on software analysis, evolution and reengineering*. Honolulu, Hawaii, USA: IEEE.

Sharif, B., Shaffer, T., Wise, J. & Maletic, J. I. (2016). Tracking Developers' Eyes in the IDE. *IEEE Software*, *33* (3), 105–108. doi: 10.1109/MS.2016.84

Sharma, L. (2016). *What is Software Development Life Cycle – SDLC?* Zugriff am 2020-08-12 auf https://www.toolsqa.com/software-testing/software-development-life-cycle/

Sheridan, H. & Reingold, E. M. (2017). The holistic processing account of visual expertise in medical image perception: A review. *Frontiers in Psychology*, *8*, 1–11. doi: 10.3389/fpsyg.2017.01620

Sillito, J., Murphy, G. C. & De Volder, K. (2006). Questions programmers ask during software evolution tasks. In *Proceedings of the 14th international symposium on foundations of software engineering (sigsoft/fse)* (S. 23). New York, New York, USA: ACM. Zugriff auf http://portal.acm.org/citation.cfm?doid=1181775.1181779 doi: 10.1145/1181775.1181779

Simon, H. A. & Chase, W. G. (1973). Skill in Chess. *American Scientist*, *61* (4), 394–403. doi: 10.1511/2011.89.106

Smith, J. D. & Graham, T. C. N. (2006). Use of eye movements for video game control. In *Proceedings of the 2006 acm sigchi international conference on advances in computer entertainment technology - ace '06* (S. 20). New York, New York, USA: ACM Press. Zugriff auf http://portal.acm.org/citation.cfm?doid=1178823.1178847 doi: 10.1145/1178823.1178847

Soloway, E. & Ehrlich, K. (1984). Empirical Studies of Programming Knowledge. *IEEE Transactions on Software Engineering*, *10* (5), 595–609. Zugriff auf http://ieeexplore.ieee.org/document/5010283/ doi: 10.1109/TSE.1984.5010283

Sommerville, I. (2012). *Software Engineering* (9th Aufl.). Munich: Pearson.

Soska, A., Mottok, J. & Wolff, C. (2016). A Study on Cognitive Deficits in Learning to Program. In *Proceedings of the 4th european conference on software engineering (ecsee)* (S. 209–214). Seeon, Deutschland. doi: 10.1787/578054332028

Spillner, A. & Linz, T. (2005). *Basiswissen Softwaretest* (3. Aufl.). Heidelberg: dpunkt.

Spinelli, L., Pandey, M. & Oney, S. (2018, apr). Attention patterns for code animations: using eye trackers to evaluate dynamic code presentation techniques. In *Conference companion of the 2nd international conference on art, science, and engineering of programming - programming'18 companion* (S. 99–104). Nice: ACM Press. Zugriff auf http://dl.acm.org/citation.cfm?doid=3191697.3214338 doi: 10.1145/3191697.3214338

SR Research. (2022). *Data Viewer.* Zugriff am 2022-04-05 auf https://www.sr-research.com/data-viewer/

Staatsministerium für Digitalisierung. (2021). *Lust auf Zukunft - aktiv, mutig & digital.* Zugriff am 2021-10-27 auf https://www.bundesregierung.de/breg-de/

bundesregierung/staatsministerin-fuer-digitalisierung

Stark, T. (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 Thesis).

Strobach, T. (2020). *Kognitive Psychologie*. Stuttgart: Kohlhammer.

Strukelj, A. & Niehorster, D. C. (2018). One page of text : Eye movements during regular and thorough reading, skimming, and spell checking. *Journal of Eye Movement Research*, *11* (1), 1–22. Zugriff auf http://dx.doi.org/10.16910/jemr.11.1.1 doi: 10.16910/jemr.11.1.1

Sweller, J. (1988). Cognitive Load During Problem Solving : Effects on Learning. *Cognitive Science*, *285* (12), 257–285.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, *4* (4), 295–312. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/0959475294900035 doi: 10.1016/0959-4752(94)90003-5

Sweller, J., van Merrienboer, J. J. & Paas, F. G. (1998). Cognitive Architecture and Instructional Design. *Educational Psychology Review*, *10* (3), 251–296. doi: 10.1023/A:1022193728205

Swensson, R. G. (1980). A two-stage detection model applied to skilled visual search by radiologists. *Perception Psychophysics*, *27* (1), 11–16. doi: 10.3758/BF03199899

Techopedia. (2022). *What does stack overflow mean?* Zugriff am 2022-06-27 auf https://www.techopedia.com/definition/9522/stack-overflow

Thomas, W. (2010). *Theoretische informatik* (Bd. 33) (Nr. 5). doi: 10.1007/s00287-010-0471-1

Thornton, S. (2016). *Avoiding stack overflow in embedded processors.* Zugriff auf https://www.microcontrollertips.com/faq-avoiding-stack-overflow-embedded-processors/

TIOBE. (2022). *TIOBE Ranking.* Zugriff am 2022-04-25 auf https://www.tiobe.com/tiobe-index/

Tobii Pro. (2020). *Tobii Spectrum Pro.* Zugriff am 2020-03-24 auf https://www.tobiipro.com/de/produkte/tobii-pro-spectrum/

Tobii Pro. (2021). *Tobii Pro Lab Description.* Zugriff am 2022-04-05 auf https://www.tobiipro.com/siteassets/tobii-pro/products/software/tobii-pro-lab/Tobii_Pro_Lab_Product_Description.pdf/?v=1.181

Tobii Technology. (2010). *Tobii T/X series Eye Trackers.* Zugriff am 2022-05-11 auf https://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-tx-product-description.pdf/?v=1.0

Torralba, A., Oliva, A., Castelhano, M. S. & Henderson, J. M. (2006). Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychological Review*, *113* (4), 766–

786. Zugriff auf http://doi.apa.org/getdoi.cfm?doi=10.1037/0033-295X.113.4.766 doi: 10.1037/0033-295X.113.4.766

Tricentis. (2017). *Software fail watch: 5th edition.* Zugriff am 2020-07-21 auf https://www.tricentis.com/resources/software-fail-watch-5th-edition/

Tufano, R., Pascarella, L., Tufano, M., Poshyvanyk, D. & Bavota, G. (2021). Towards Automating Code Review Activities. *Proceedings of the International Conference on Software Engineering*, 163–174. Zugriff auf http://arxiv.org/abs/2101.02518 doi: 10.1109/ICSE43902.2021.00027

Turner, R., Falcone, M., Sharif, B. & Lazar, A. (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)* (S. 231–234). Safety Harbor, Floriada, USA: ACM. Zugriff auf http://doi.acm.org/10.1145/2578153.2578218 doi: 10.1145/2578153.2578218

Universität Regensburg. (2024). *Webseite der Fakultät für Informatik und Data Science.* Zugriff am 2024-02-05 auf https://www.uni-regensburg.de/informatik-data-science/fakultaet/startseite/index.html

Unkelos-Shpigel, N. & Hadar, I. (2015). Gamifying software engineering tasks based on cognitive principles: The case of code review. *Proceedings - 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015*, 119–120. doi: 10.1109/CHASE.2015.21

Uwano, H., Nakamura, M., Monden, A. & Matsumoto, K.-i. (2006). Analyzing Individual Performance of Source Code Review Using Reviewers' Eye Movement. In *Proceedings of the symposium on eye tracking research and applications (etra)* (S. 133–140). San Diego, CA: ACM. doi: http://doi.acm.org/10.1145/1117309.1117357

van der Hoek, J. (2018). *Pursuing a Full Agile Software Development Life Cycle.* Zugriff auf https://www.mendix.com/blog/pursuing-a-full-agile-software-lifecycle/

van de Wiel, M. W. J., van den Bossche, P., Janssen, S. & Jossberger, H. (2011). Exploring deliberate practice in medicine: How do physicians learn in the workplace? *Advances in Health Sciences Education*, *16* (1), 81–95. doi: 10.1007/s10459-010-9246-3

van Gog, T., Ericsson, K. A., Rikers, R. M. J. P. & Paas, F. (2005). Instructional design for advanced learners: Establishing connections between the theoretical frameworks of cognitive load and deliberate practice. *Educational Technology Research and Development*, *53* (3), 73–81. Zugriff auf http://link.springer.com/10.1007/BF02504799{%}5Cnhttp://www.scopus.com/inward/record.url?eid=2-s2.0-23744475019{&}partnerID=40{&}md5=e006fba5d69981d7897cee45f98b6b1d doi: 10.1007/BF02504799

Voisin, S., Pinto, F., Xu, S., Morin-Ducote, G., Hudson, K. & Tourassi, G. D. (2013). Investigating the association of eye gaze pattern and diagnostic error in mammography. In C. K. Abbey & C. R. Mello-Thoms

(Hrsg.), *Medical imaging* (S. 867302). Lake Buena Vista, FL, USA. Zugriff auf http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2007908 doi: 10.1117/12.2007908

Weinberg, G. M. & Schulman, E. L. (1974). Goals and Performance in Computer Programming. *Human Factors:*, *16* (1), 70–77. Zugriff auf http://journals.sagepub.com/doi/10.1177/001872087401600108 doi: 10.1177/001872087401600108

Williams, A. M., Ford, P. R., Hodges, N. J. & Ward, P. (2018). Expertise in Sport: Specificity, Plasticity, and Adaptability in High-Performance Athletes. In *The cambridge handbook of expertise and expert performance* (S. 653–674). Cambridge University Press. Zugriff auf https://www.cambridge.org/core/product/identifier/9781316480748%23CN-bp-34/type/book_part doi: 10.1017/9781316480748.034

Witte, F. (2016). *Testmanagement und Softwaretest: Theoretische Grundlagen und praktische Umsetzung*. Wiesbaden, Deutschland: Springer Fachmedien Wiesbaden. Zugriff auf http://link.springer.com/10.1007/978-3-8348-2142-3_3http://link.springer.com/10.1007/978-3-658-09964-0 doi: 10.1007/978-3-658-09964-0

Wittkowski, K. M. (1988, dec). Friedman-Type Statistics and Consistent Multiple Comparisons for Unbalanced Designs with Missing Data. *Journal of the American Statistical Association*, *83* (404), 1163. Zugriff auf https://www.jstor.org/stable/2290150?origin=crossref doi: 10.2307/2290150

Wolfe, J. M., Võ, M. L.-H., Evans, K. K. & Greene, M. R. (2011). Visual search in scenes involves selective and nonselective pathways. *Trends in Cognitive Sciences*, *15* (2), 77–84. Zugriff auf https://linkinghub.elsevier.com/retrieve/pii/S1364661310002536 doi: 10.1016/j.tics.2010.12.001

Wolff, C. E., Jarodzka, H. & Boshuizen, H. P. (2017). See and tell: Differences between expert and novice teachers' interpretations of problematic classroom management events. *Teaching and Teacher Education*, *66*, 295–308. Zugriff auf http://dx.doi.org/10.1016/j.tate.2017.04.015 doi: 10.1016/j.tate.2017.04.015

Wood, B. P. (1999). Visual Expertise. *Radiology*, *211* (1), 1–3. Zugriff auf http://pubs.rsna.org/doi/10.1148/radiology.211.1.r99ap431 doi: 10.1148/radiology.211.1.r99ap431

Wood, G., Knapp, K. M., Rock, B., Cousens, C., Roobottom, C. & Wilson, M. R. (2013). Visual expertise in detecting and diagnosing skeletal fractures. *Skeletal Radiology*, *42* (2), 165–172. doi: 10.1007/s00256-012-1503-5

Wunderlich-Pfeiffer, F. (2015). *Softwarefehler in der Raumfahrt: In den Neunzigern stürzte alles ab.* Zugriff am 2020-07-21 auf https://www.golem.de/news/softwarefehler-in-der-raumfahrt-in-den-neunzigern-stuerzte-alles-ab-1511-117537.html

Zemblys, R., Niehorster, D. C. & Holmqvist, K. (2019). gazeNet: End-to-end eye-

movement event detection with deep neural networks. *Behavior Research Methods*, *51* (2), 840–864. doi: 10.3758/s13428-018-1133-5

Zemblys, R., Niehorster, D. C., Komogortsev, O. & Holmqvist, K. (2018). Using machine learning to detect events in eye-tracking data. *Behavior Research Methods*, *50* (1), 160–181. doi: 10.3758/s13428-017-0860-3

.1 Fragebögen

.1.1 Einwilligungserklärung zur C-Studie

Die Einwilligungserklärung zur C-Studie bzw. der integrierten Fragebogen findet sich auf den nachfolgenden Seiten.

.1.2 Einwilligungserklärung zur C++-Studie

Die Einwilligungserklärung zur C++-Studie bzw. der integrierten Fragebogen findet sich auf den nachfolgenden Seiten.

.2 Verwendete Stimuli

.2.1 C-Codebeispiele

Die für die C-Studie erstellten Codebeispiele befinden sich im digitalen Anhang auf dem beigefügten USB-Stick.

.2.2 C++-Codebeispiele

Die für die C++-Studie erstellten Codebeispiele befinden sich im digitalen Anhang auf dem beigefügten USB-Stick.

.3 Rohdaten und R-Skripte

.3.1 Rohdaten und R-Skripte zur C-Studie

Die im Rahmen der C-Studie erhobenen Daten finden sich im digitalen Anhang dieser Arbeit. Alle Daten befinden sich in einem anonymisierten und pseudonymisierten Format.
Die zur Datenanalyse verwendeten R-Skripte finden sich ebenfalls auf dem beigelegten USB-Stick.

.3.2 Rohdaten und R-Skripte zur C++-Studie

Die im Rahmen der C++-Studie erhobenen Daten finden sich im digitalen Anhang dieser Arbeit. Alle Daten befinden sich in einem anonymisierten und pseudonymisierten Format.
Die zur Datenanalyse verwendeten R-Skripte finden sich ebenfalls auf dem beigelegten USB-Stick.

Sehr geehrte Versuchsteilnehmerin, Sehr geehrter Versuchsteilnehmer,

vielen Dank, dass Sie sich für die Teilnahme an dieser Studie entschieden haben.
Durch diesen Versuch soll geklärt werden, wie Programmierer einen Quellcode lesen und welche kognitiven Prozesse dabei involviert sind.
Dazu werden Ihnen nachfolgend sechs Fallbeispiele gezeigt, die sie durchlesen und auf ihre Korrektheit hin untersuchen sollen. All Ihre Augenbewegungen werden durch einen Eye-Tracker aufgezeichnet und im Anschluss ausgewertet. Zusätzlich zu dieser Analyse sind ergänzende Interviews an einem weiteren Termin geplant. Dabei sollen Auffälligkeiten in Ihren Augenbewegungen besprochen und vertieft werden. Eine Teilnahme an diesen erfolgt auf freiwilliger Basis und kann individuell mit Ihnen abgestimmt werden.
Alle gesammelten Daten werden von uns anonym und mit größter Sorgfalt behandelt.

Bitte geben Sie nachfolgend noch folgende Daten an:

Emailadresse:
_____

Alter:
_____

Geschlecht:
_____

Erfahrungen als Programmierer:
_____
_____
_____

Wie viele Jahre programmieren Sie bereits:
_____

Abbildung 1: Einwilligungserklärung zur C-Studie (Seite 1)

Wie viele Jahre haben Sie bereits Berufserfahrung als Programmierer:

_____

Schätzen Sie Ihre allgemeinen Programmierkenntnisse als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu
1	2	3	4

Schätzen Sie Ihre Programmierkenntnisse in C als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu
1	2	3	4

Schätzen Sie Ihre Fähigkeiten als Reviewer als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu
1	2	3	4

Durch Ihre Unterschrift bestätigen Sie, dass Sie die oben genannten Informationen gelesen haben, mit diesen einverstanden sind und freiwillig an dieser Studie teilnehmen.

_____ _____

Ort und Datum Unterschrift

Abbildung 2: Einwilligungserklärung zur C-Studie (Seite 2)

Sehr geehrte Versuchsteilnehmerin, sehr geehrter Versuchsteilnehmer,

vielen Dank, dass Sie sich für die Teilnahme an dieser Studie entschieden haben.
Durch diesen Verusch soll geklärt werden, wie Programmierer einen Quellcode reviewen und welche kognitiven Prozesse dabei involviert sind.
Dazu werden Ihnen nachfolgend sechs Fallbeispiele gezeigt, die sie durchlesen und auf ihre Korrektheit hin untersuchen sollen. All Ihre Augenbewegungen werden durch einen Eye-Tracker aufgezeichnet und im Anschluss ausgewertet. Zusätzlich zu dieser Analyse sind ergänzende Interviews geplant. Dabei sollen Auffälligkeiten in Ihren Augenbewegungen besprochen und vertieft werden.

Bitte geben Sie nachfolgend noch folgende Daten an:

Emailadresse:

_____

Alter:

_____

Geschlecht:

_____

Beruf:

_____

Erfahrungen als Programmierer:

_____

_____

_____

Wie viele Jahre programmieren Sie bereits:

_____

Wie viele Jahre haben Sie bereits Berufserfahrung als Programmierer:

_____

Abbildung 3: Einwilligungserklärung zur C++-Studie (Seite 1)

Schätzen Sie Ihre allgemeinen Programmierkenntnisse als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu

Schätzen Sie Ihre Programmierkenntnisse in C++ als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu

Schätzen Sie Ihre Fähigkeiten als Reviewer als gut ein?

Trifft gar nicht zu	Trifft eher nicht zu	Trifft eher zu	Trifft voll und ganz zu

Durch Ihre Unterschrift bestätigen Sie, dass Sie die oben genannten Informationen gelesen haben, mit diesen einverstanden sind und freiwillig an dieser Studie teilnehmen. Weiterhin erlauben Sie dem LaS³ die von Ihnen erhobenen Daten in anonymisierter Form zu speichern und im Laufe einer wissenschaftlichen Studie auszuwerten.

Regensburg, den _____ _____

Unterschrift des Probanden

Abbildung 4: Einwilligungserklärung zur C++-Studie (Seite 2)

Eidesstattliche Erklärung

Hiermit erkläre ich, Florian Hauser (geboren am 19. Oktober 1990),

an Eides statt gegenüber der Fakultät für Humanwissenschaften der Universität Regensburg, dass ich die vorliegende Dissertation mit dem Titel *Visuelle Expertise bei Code Reviews* selbstständig und nur unter Zuhilfenahme der im Quellen- und Literaturverzeichnis genannten Werke angefertigt habe.

Regensburg, den 1. April 2024

Florian Hauser (M.A.)