Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Entwicklung eines Prototyps zur Filterung von Trace Links basierend auf Eye Tracking Daten

Bachelorarbeit

im Studiengang Informatik

von

Tim Heitmann

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüferin: Dr. Jil Klünder
Betreuerin: M.Sc. Maike Ahrens

Hannover, 07. August 2023

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 07.08.2023

_____

Tim Heitmann

Zusammenfassung

Entwicklung eines Prototyps zur Filterung von Trace Links basierend auf Eye Tracking Daten

Requirements Traceability kann in der Software Entwicklung ein nützliches Tool sein, um sowohl die Entwicklung als auch das Projektmanagement effizienter zu gestalten, indem Trace Links zwischen Artefakten der Entwicklung erstellt und gepflegt werden. Das manuelle Erstellen der Links ist sehr zeitaufwendig und aktuelle Softwarelösungen, die auf Information Retrieval oder auf Machine Learning basieren, weisen ebenfalls Probleme auf.
In dieser Arbeit soll ein weiterer Ansatz verfolgt werden und untersucht werden, ob Trace Links automatisiert aus Eye Tracking Aufnahmen erzeugt werden können. Hierzu wird ein Prototyp konzeptioniert und entwickelt, der zuvor aufbereitete Eye Tracking Daten analysiert. Die Aufbereitung der Daten besteht dabei aus dem Definieren von Areas of Interest (AOI), zwischen denen die Links generiert werden. Für die Generierung der Links werden eine Reihe an Heuristiken definiert, die verschiedene Parameter der Eye Tracking Aufnahme berücksichtigen und abhängig davon Trace Links erzeugen.
Die Qualität der Links, die der in diese Arbeit entwickelte Prototyp generiert, kommt zwar nicht an die der bisherigen Lösungen heran. Dennoch werden durch den Entwicklungsprozess und die anschließende Evaluation Probleme klar, die es auf dem Weg hin zu einer Generierung von Trace Links basierend auf Eye Tracking Daten zu lösen gilt.

Abstract

Development of a Prototype for Filtering Trace Links based on Eye Tracking Data

Requirements traceability can be a useful tool in software development to make both development and project management more efficient by creating and maintaining trace links between development artefacts. The manual creation of links is very time-consuming and current software solutions based on information retrieval or machine learning also show some issues. In this work, a further approach will be pursued and it will be investigated whether trace links can be generated automatically from eye tracking recordings. For this purpose, a prototype will be conceptualised and developed, that analyses prepared eye tracking data. The processing of the data consists of defining Areas of Interest (AOI) between which the links are generated. For the generation of the links, a series of heuristics are defined that take into account various parameters of the eye tracking recording and generate trace links depending on them.

The quality of the links generated by the prototype developed in this work does not match that of previous solutions. Nevertheless, the development process and the subsequent evaluation reveal problems that need to be solved on the way to generating trace links based on eye tracking data.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Requirements Traceability – das ist die Fähigkeit, Anforderungen über den gesamten Lebenszyklus einer Software hinweg nachverfolgen zu können, und ist somit ein nützliches Tool in der Softwareentwicklung. Dabei werden sogenannte Trace Links zwischen den Requirements und Artefakten hergestellt, die während der Entwicklung entstehen, wie z.B. Codeabschnitte, Dokumentationen oder Tasks. Diese Links können im Anschluss genutzt werden, um ein effizientes Projekt Management zu ermöglichen, da Zusammenhänge schneller erkannt werden können und der Aufwand für das Umsetzen von Requirements besser abgeschätzt werden kann, aber auch um die Qualität der Software zu gewährleisten, indem Fehler innerhalb des Quellcodes schneller behoben werden können, wenn nachzuvollziehen ist, welche anderen Artefakte ebenfalls betroffen sein können [1].

Das manuelle Erstellen der Trace Links ist jedoch sehr personal- und zeitaufwändig. Es existiert bereits Software, die unterstützend eingesetzt werden kann. Diese setzen meist auf einen Machine Learning Ansatz oder nutzen ein Information Retrieval-System (IRS) [2]. Beim Machine Learning Ansatz werden zunächst mögliche Trace Links durch eine künstliche Intelligenz (KI) generiert [3]. Um korrekt zu funktionieren, muss die KI jedoch intensiv trainiert werden, wofür in der Regel nicht ausreichend Daten zur Verfügung stehen, sodass dieser Ansatz nicht effektiv genutzt werden kann.

Anwendungen, die ein IRS verwenden, erzeugen die Links aus der Dokumentation der Requirements und der Design Elemente. Hayes et al. [4] verordnen in diesem Ansatz vor allem Probleme bei der Dokumentation. Diese ist oft unvollständig oder wird von verschiedenen Stakeholdern in unterschiedlichem Jargon geschrieben, sodass eine eindeutige Zuordnung nicht möglich ist.

In dieser Arbeit soll daher ein weiterer Ansatz untersucht werden: Das Verwenden von Eye Tracking Daten. Diese werden durch einen Eye Tracker aufgenommen und beinhalten die genauen Blickpunkte des Nutzers inklusive Zeitstempel. Kombiniert mit einer zeitgleichen Bildschirmaufnahme lässt

sich so im Nachhinein ermitteln, wann die Person auf welches Element auf dem Bildschirm geschaut hat. Zurzeit wird Eye Tracking häufig für Analysen von Benutzeroberflächen oder Websites verwendet, indem in den während der Nutzung aufgenommenen Daten nach Mustern gesucht wird, um Aussagen darüber treffen zu können, welche Bereiche der Nutzeroberfläche besonders oft angeschaut wurden. Somit lassen sich Elemente anpassen, die zu viel oder zu wenig Aufmerksamkeit beim Nutzer erregen, um intuitiv bedienbar zu sein.

Aber auch im Bereich des Software Engineerings wird diese Technologie bereits genutzt. So kann beispielsweise veranschaulicht werden, wie Entwickler Code lesen und verstehen, oder während der Entwicklung Änderungen am Quellcode parallel getrackt werden, um nachvollziehen zu können, wie sich einzelne Codeabschnitte verändern.

1.2. Zielsetzung

Im Rahmen dieser Arbeit soll daher eine Anwendung entwickelt werden, die es erlaubt aus Eye Tracking Daten eine Liste potentieller Links zwischen Elementen auf dem Bildschirm zu erstellen. Angewendet werden soll dies primär auf Blickdaten von Entwicklern und anderen Stakeholdern im Software Entwicklungsprozess, um diese beim Nachvollziehen der Anforderungen innerhalb des Softwareprojekts zu unterstützen. Um repräsentative Daten zu erhalten, sollen mit mehrere Entwickler Eye Tracking Aufnahmen durchgeführt werden, während diese an ihren jeweiligen Tasks arbeiten. Anhand dieser Daten werden im Laufe der Arbeit Heuristiken entwickelt, die es ermöglichen sollen, aus den Eye Tracking Daten der Entwickler potentielle Trace Links herzustellen.

Der Prototyp soll zudem über ein benutzerfreundliches User Interface einerseits die Eingabe von Eye Tracking Daten, andererseits aber auch die intuitive Auswahl und Konfiguration von Heuristiken ermöglichen. Zudem müssen die Ergebnisse übersichtlich dargestellt werden. Die Heuristiken sind Regelsätze, wie z.B. ein maximaler zeitlicher Abstand zwischen dem Betrachten von zwei Elementen oder die Häufigkeit der Blicksprünge zwischen diesen. Während

der Entwicklung der Anwendung soll neben Korrektheit und Testbarkeit auch auf eine gute Wartbarkeit geachtet werden, sodass nach Abschluss der Arbeit weitere Heuristiken in das Programm eingepflegt werden können.

1.3. Struktur der Arbeit

Die vorliegende Arbeit ist wie folgt gegliedert. Im folgenden Kapitel 2 werden grundlegende Technologien und Begriffe erläutert, die für diese Arbeit relevant sind. Anschließend wird in Kapitel 3 auf verwandte Arbeiten eingegangen und eine Abgrenzung zu dieser Arbeit vorgenommen. Im 4. Kapitel wird eine Anforderungsanalyse durchgeführt und im darauffolgenden Kapitel 5 ein Konzept entwickelt. Das 6. Kapitel behandelt einige Details der Implementierung der Software. Die erzeugten Links werden in Kapitel 7 evaluiert und eine Analyse möglicher Probleme vorgenommen. Abschließend wird im Kapitel 8 ein Fazit gezogen und auf mögliche Erweiterungen der Anwendung eingegangen.

2. Grundlagen

In diesem Kapitel werden zunächst grundlegende Begriffe erläutert. So wird zuerst auf die Technologie „Eye Tracking“ eingegangen und anschließend auf das Konzept „Requirements Traceability“.

2.1. Eye Tracking

Dieser Abschnitt gibt einen Überblick, was Eye Tracking ist und in welchen Bereichen diese Technologie Anwendung findet.

2.1.1. Was ist Eye Tracking?

Eye Tracking ist eine Technologie und Methodik, mit der die Blickbewegungen und -muster einer Person erfasst und analysiert werden können. Diese Technologie hat in den letzten Jahren aufgrund ihrer vielfältigen Anwendungsmöglichkeiten und ihres Potenzials zur Erforschung menschlichen Verhaltens und visueller Informationsverarbeitung stark an Bedeutung gewonnen.

Eye Tracking basiert auf der Erfassung der Bewegungen des Augapfels und/oder der Pupille. Es gibt verschiedene Ansätze und Technologien, die bei der Umsetzung von Eye Tracking zum Einsatz kommen. Einer der gängigsten Ansätze verwendet Infrarotlicht, das auf die Augen des Betrachters gerichtet wird. Spezielle Kameras erfassen dann die Reflexionen des Infrarotlichts auf der Hornhaut oder der Pupille, um die Position und Bewegung der Augen zu verfolgen. Auf diese Weise können Blickrichtung und Blickverlauf bestimmt werden. Die Aufnahmen, die in dieser Arbeit verwendet werden, wurden mit dem Tobii Pro X3-120 aufgezeichnet. Dieser verfügt über einen Infrarot-Sender und zwei Infrarot-Kameras, die mit einer Frequenz von 120 Hz das von den Hornhäuten des Nutzers reflektierte Licht aufnehmen.

Abbildung 1- Ein Eye Tracker der Marke Tobii (Quelle: https://imagingsol.eu/product/tobii-pro-x3-120/)

Die gesammelten Daten werden dann mit einer speziellen Software ausgewertet. In der Regel erfolgt zu Beginn einer Aufnahme eine Kalibrierung, bei der der Betrachter auf verschiedene Punkte auf dem Bildschirm schaut, um die Blickdaten mit den tatsächlichen Positionen zu verknüpfen. Ist die Kalibrierung abgeschlossen, kann das System die Blickdaten genauen Positionen auf dem Bildschirm zuordnen.

Die Auswertung der Eye-Tracking-Daten umfasst verschiedene Parameter wie Blickposition oder die Art der Blickbewegung. Hierbei wird zwischen Fixationen, was ein längeres Verharren des Blickes und somit das genaue Betrachten eines Punktes beschreibt, und Sakkaden unterschieden, was eine flüchtige Augenbewegung darstellt, wie zum Beispiel beim Lesen eines Textes. Aus der Analyse dieser Parameter lassen sich verschiedene Aspekte des Blickverhaltens ableiten, wie z.B. Aufmerksamkeitsmuster, visuelle Informationsverarbeitung und Reaktionszeiten auf visuelle Reize, die sich in einer Vielzahl von Anwendungsbereichen nutzen lassen.

2.1.2. Anwendungsbereiche des Eye Trackings

Ein großes Anwendungsgebiet von Eye Tracking ist die Verbesserung der Usability und User Experience von Produkten, Websites und digitalen Anwendungen. Durch die genaue Verfolgung der Blickbewegungen können Designer Schwachstellen in der Benutzeroberfläche identifizieren und entsprechende Optimierungen vornehmen, um eine intuitive und effiziente Benutzerführung zu gewährleisten. Für solche Analysen wird aus Eye Tracking aufnahmen, die während der Nutzung der Software aufgezeichnet werden, eine sogenannte Heatmap erstellt, auf der klar ersichtlich ist, auf welche Elemente

der Nutzer besonders häufig schaut. So können beispielsweise besonders ablenkende Designelemente überarbeitet und die User Experience verbessert werden [5].

Abbildung 2- Beispiel einer Heatmap (Quelle: https://upload.wikimedia.org/wikipedia/commons/5/50/Eyetracking_heat_map_Wikipedia.jpg)

Im Marketing liefert Eye Tracking wertvolle Informationen über die visuelle Wahrnehmung von Konsumenten. Durch die Analyse der Blickbewegungen können Unternehmen nachvollziehen, welche Elemente ihrer Werbung am meisten beachtet werden und wie visuelle Gestaltungselemente die Aufmerksamkeit der Betrachter beeinflussen. Dies kann sowohl bei Werbekampagnen im Internet, aber auch für Werbung im sonstigen Alltag, wie zum Beispiel im Supermarkt oder auf Werbetafeln, verwendet werden [5].

Ein weiteres großes Anwendungsgebiet von Eye Tracking liegt in der medizinischen Forschung. Es kann zur Diagnose und Überwachung neurologischer Erkrankungen wie Schlaganfall, Parkinson oder Aufmerksamkeitsdefizit-/Hyperaktivitätsstörung (ADHS) eingesetzt werden [6]. Durch die detaillierte Analyse der Blickbewegungen können Ärzte und Forscher frühzeitig Parkinson diagnostizieren und von ähnlichen Krankheitsbildern abgrenzen, da sich die Geschwindigkeit während der Sakkaden stark unterscheidet [7]. Auch ADHS kann mittels Eye Tracking zuverlässig diagnostiziert werden. Lev et al. [8] haben hierfür die Blickbewegungen von ADHS-Patienten mit den Blickbewegungen einer Kontrollgruppe verglichen. Dabei zeichnet sich

ein Muster ab, dass ADHS-Patienten mehr dazu neigen auf unwichtige Elemente zu schauen und sogar häufiger den Blick ganz vom Monitor abwenden. Auch in der Softwareentwicklung kommt Eye Tracking bereits zum Einsatz und dient hier zum Verstehen von Prozessen und Abläufen [9]. Sharif und Shaffer heben unter anderem Tools zur Unterstützung von Entwicklern beim beheben von Bugs oder zur Verbesserung der Readability von Code hervor [10]. Die genannten Anwendungsbereiche stellen nur eine Auswahl der vielfältigen Möglichkeiten von Eye Tracking dar.

2.2. Requirements Traceability

Requirements Traceability ist ein Werkzeug, um den Lebenszyklus von Anforderungen nachzuvollziehen und deren Verbindung zu anderen Elementen im Entwicklungsprozess aufrechtzuerhalten. Dabei handelt es sich um einen systematischen Ansatz, der sicherstellt, dass jede Anforderung von ihrer Entstehung bis zur Erfüllung verfolgt werden kann, indem die Beziehungen zwischen Anforderungen, Design, Implementierung, Testfällen und anderen Artefakten dokumentiert und verwaltet werden.

Requirements Traceability ist aus mehreren Gründen wichtig. Erstens ermöglicht es die Rückverfolgbarkeit von Anforderungen, so dass Entwickler und andere Projektbeteiligte den Ursprung und die Entwicklung von Anforderungen nachvollziehen können. Dies fördert eine effektive Kommunikation, Koordination und Konsistenz innerhalb des Projektteams [11].

Zweitens erleichtert Requirements Traceability das Änderungsmanagement. Wenn Anforderungen im Laufe des Projekts geändert werden müssen, kann Traceability helfen, die Auswirkungen dieser Änderungen zu verstehen. Es wird deutlich, welche anderen Artefakte betroffen sind und wie diese angepasst werden müssen.

Darüber hinaus ermöglicht Traceability eine vollständige Testabdeckung. Durch die Verknüpfung der Anforderungen mit den entsprechenden Testfällen kann sichergestellt werden, dass alle Anforderungen getestet werden. Dies verbessert die Qualitätssicherung, da keine Anforderung übersehen wird.

Ein weiterer wichtiger Aspekt ist das Debugging. Treten Fehler im System auf, hilft die Requirements Traceability, die Ursachen zu ermitteln und die

betroffenen Anforderungen zu identifizieren. Dies ermöglicht eine gezielte Fehlerbehebung und erleichtert das Verständnis der Auswirkungen von Fehlerkorrekturen.

Requirements Traceability kann in verschiedenen Phasen des Softwareentwicklungsprozesses eingesetzt werden. Am Anfang steht die Anforderungserfassung, bei der die Anforderungen identifiziert, dokumentiert und verfolgt werden, um sicherzustellen, dass sie vollständig und konsistent sind. Während des Designprozesses wird Traceability verwendet, um die Verbindung zwischen den Anforderungen und den entsprechenden Designelementen herzustellen. In der Implementierungsphase ermöglicht die Traceability die Verfolgung der Umsetzung der Anforderungen im Quellcode, um sicherzustellen, dass alle Anforderungen erfüllt werden. Während des Testprozesses wird die Traceability zur Erstellung von Testfällen verwendet, um sicherzustellen, dass alle Anforderungen abgedeckt sind. Im Falle von Anforderungsänderungen kann die Traceability verwendet werden, um die Auswirkungen auf andere Bereiche des Projekts zu analysieren und Anpassungen vorzunehmen.

Requirements Traceability ist daher ein mächtiges Werkzeug im Softwareentwicklungsprozess. Es ermöglicht eine klare Verfolgung der Anforderungen und ihrer Beziehungen untereinander und zu anderen Elementen des Projekts, verbessert die Kommunikation und Koordination im Team, erleichtert das Änderungsmanagement und erhöht die Qualitätssicherung [12].

3. Verwandte Arbeiten

In diesem Kapitel wird zunächst näher darauf eingegangen, welche Ansätze bereits genutzt werden, um das Traceability Problem, das von Gotel und Finkelstein [13] und von Arkley und Riddle [14] beschrieben wird, zu lösen und welche Probleme diese Ansätze selbst hervorbringen. Anschließend werden einige Arbeiten vorgestellt, die sich sowohl theoretisch als auch praktisch mit dem Verknüpfen von Eye Tracking und Requirements Traceability befassen. Abschließend erfolgt eine Abgrenzung dieser Arbeit zu den verwandten Arbeiten.

3.1. Bisherige Ansätze von Requirements Traceability

Das Erzeugen von Trace Links durch Softwarelösungen basiert in den meisten Fällen entweder auf Machine Learning oder es wird ein Information Retrieval-System (IRS) verwendet. Diese beiden Ansätze wurden bereits in einigen wissenschaftlichen Arbeiten auf Funktionsweise, aber auch auf mögliche Probleme untersucht, auf die in den folgenden Abschnitten eingegangen wird.

3.1.1. Information Retrieval-Systeme

Information Retrieval (IR) ist ein Feld der Informatik, das das Problem behandelt, relevante Dokumente in einer großen Dokumentensammlung zu finden. Die im IR am meisten untersuchte Methodik ist keyword-based Retrieval. Dabei wird jedes Dokument analysiert und Schlagwörter ermittelt, die für dieses Dokument wichtig sind und anschließend mit dem Dokument verknüpft werden. Durch User Queries kann dann aus der Sammlung ein gewünschtes Dokument herausgesucht werden [4].

Hayes et al. [4] untersuchen in ihrer Arbeit eine Reihe von IR-Algorithmen. Die Algorithmen sind zum einen ein gewöhnlicher IR-Algorithmus, der Schlagworte erzeugt und gewichtet, zum anderen zwei erweiterte Algorithmen. Der erste erweiterte Algorithmus berücksichtigt zusätzlich Feedback der Nutzer zu den Ergebnissen aus dem gewöhnlichen IR-Algorithmus. Der zweite erweiterte Algorithmus, in der Arbeit „Thesaurus-based“ genannt,

berücksichtige zudem auch die Synonyme der Schlagwörter, die in einer Art Wörterbuch zusammengetragen sind.

Für die Evaluation der Genauigkeit der Algorithmen wurden zwei Datensätze aus high-level Requirements und low-level Requirements zusammengestellt, die sowohl von Analysten als auch von den Algorithmen und einem weiteren Requirements Tracing Tool bewertet wurden. Es zeigte sich, dass der Thesaurus-base Algorithmus zwar einen Großteil der Links finden konnte (85,36%), jedoch auch sehr viele inkorrekte Links erzeugt hat und somit nur eine Genauigkeit von 40,69%a aufweisen konnte. Probleme bei diesem Ansatz sehen Hayes et al. bei der Größe der Datensätze, die mit 69 Elementen deutlich kleiner sind als Dokumentensammlungen, für die IRS sonst eingesetzt werden, und bei den unvollständigen und zweideutigen Requirement Dokumenten.

Borg und Runeson [2] führten zum Thema IR in Software Traceability eine Mapping Studie durch und vergleich 79 Veröffentlichungen, die sich mit der Nutzung von IR-Algorithmen zum Zwecke des Erzeugens von Trace Links befassen. In 25 dieser Veröffentlichungen werden für die jeweils verwendeten IR-Algorithmen Werte für Precision und Recall genannt. Der Vergleich der Ergebnisse der Veröffentlichungen zeigte, dass keine empirische Evidenz vorliegt, dass ein IR-Algorithmus anderen überlegen sei. Es wird jedoch hervorgehoben, dass es üblich ist einfache IR-Algorithmen zu verwenden, wie der, der von Hayes et al. untersucht wurde.

Ali et al. [15] kombinieren den IR-Ansatz bereits mit Eye Tracking Aufnahmen, indem sie mit einer empirischen Studie untersuchen, wie Entwickler Trace Links bewerten. In der Studie werden mit 24 Entwickler Eye Tracking Aufnahmen durchgeführt, während sie Trace Links bewerten. Die Ergebnisse zeigen, dass sich die Herangehensweise von allen Entwicklern unterschieden hat im Bezug auf Präferenzen bei Klassennamen, Methodennamen und Kommentaren. Dies adressiert das mögliche Problem, dass die Verhaltensweisen verschiedener Entwickler sich zu stark unterscheiden, um dieses durch Heuristiken abbilden zu können.

3.1.2. Machine Learning

Li et al. [3] untersuchen die Genauigkeit einer künstlichen Intelligenz zur Erzeugung von Trace Links. Hierzu wurde ein zweistufiges Modell erstellt, das im ersten Schritt durch Machine Learning anhand von Traingsdaten potentielle Links erstellt. Diese Links werden anschließend durch eine Logik und eine Reihe an Regeln erweitert. Diese Regeln verknüpfen beispielsweise bei mehreren Code Abschnitten, die mit einander verknüpft sind auch die dazu gehörenden Use Cases mit den übrigen Code Abschnitten.
Für die Evaluation wurden in dieser Arbeit mehrere bereits bewertete Datensätze verwendet, aus denen das beschriebene Modell Trace Links erstellt hat. Auch hier wurden die Precision und Recall Werte bestimmt und mit gängigen IRS verglichen. Es zeigt sich, dass das verwendete Modell eine bedeutend höhere Präzision aufweist und somit weniger falsche Links generiert. Der Recall Wert hingegen scheint sich kaum von denen eines IRS zu unterscheiden.

3.2. Eye Tracking und Requirements Traceability

Mit der Thematik, aus Eye Tracking Daten Trace Links zu generieren, haben sich bisher einige wissenschaftliche Arbeiten beschäftigt. Während einige dabei lediglich theoretische Konzepte behandeln, wurden in anderen Arbeiten bereits Tools entwickelt, deren Ziel es ist, das Traceability Problem mit Eye Tracking Daten zu lösen.
Mit iTrace haben Shaffer et al. [16] im Rahmen einer wissenschaftlichen Arbeit ein Plugin für Eclipse entwickelt. Das Konzept des Plugins wurde bereits in einer Arbeit zwei Jahre zuvor dargestellt [17]. Dieses Plugin dient zum Aufzeichnen von Blickbewegungen innerhalb der IDE und soll es ermöglichen, Eye Tracking Studien auf großen Software Systemen durchzuführen. Das Plugin trackt zunächst in den Quellcode Dateien einzelne Software Artefakte, wie beispielsweise Enumerationen oder Schleifen, und gibt diese anschließend als JSON- oder XML-Datei aus. Auch die Blickdaten werden aufgezeichnet und ausgegeben. Das Plugin selbst erzeugt keine Trace Links, wurde jedoch bereits in weiteren Arbeiten zum Aufnehmen der Daten verwendet.

Eine dieser Arbeiten stammt von Walters et al. [18]. Im Rahmen dieser Arbeit wurde ein Algorithmus, genannt SimpleGraph Gaze-Link Algorithmus, entwickelt, der basierend auf den Eye Tracking Daten von iTrace Trace Links ausgibt. Der Algorithmus stellt also lediglich Links zwischen Code Artefakten her. Für die Berechnung der Links stehen neben den Blickdaten auch der eigentliche Quellcode zur Verfügung, sodass beides in die Berechnung einfließen kann. Nach der Entwicklung des Algorithmus wurden die erzeugten Links anderen Entwicklern vorgelegt, um diese bewerten zu lassen. Es zeigt sich, dass die Präzision des entwickelten Algorithmus mit einem Maximum von 33% in folgenden Arbeiten noch verbessert werden muss.
Sharif et al. [19] entwickeln den SimpleGraph Gaze-Link Algorithmus in ihrer Arbeit weiter und vergleichen ihm mit gängigen IRS. Fünf Entwickler wurden in der Studie gebeten, acht Bugs zu beheben. Währenddessen wurden die Entwickler mit dem iTrace Plugin aufgezeichnet, um Trace Links generieren zu können. Es zeigt sich, dass der erweiterte Algorithmus in vielen Fällen bessere Ergebnisse erzeugt als die üblichen IRS, was zeigt, dass Eye Tracking Daten das Potential haben, dazu genutzt werden zu können, Trace Links zu erzeugen. Dies bestätigen auch Sharif et al. [20] mit einer empirischen Studie

3.3. Abgrenzung der Arbeit

Viele der oben beschrieben Arbeiten stellen Algorithmen vor, die neben den Eye Tracking Daten auch noch andere Daten zum Erzeugen der Links verwenden. So analysiert der SimpleGraph Ganz-Link Algorithmus von Walters et al. neben den Eye Tracking Daten auch noch den Code an sich, um Zusammenhänge zu finden. Wiederum andere Arbeiten nutzen einen gänzlich anderen Ansatz und generieren Trace Links durch Machine Learning oder IR. Alle diese Vorgehensweisen benötigen eine große Menge an Daten und sind meist nur schwer in das Arbeitsumfeld eines Entwicklers einzubinden. Mit eine einfachen Eye Tracking Aufnahme können die Eingabedaten hingegen ohne viel Aufwand erzeugt werden und umfassen einen deutlich geringeren Umfang als die Daten, die die anderen Ansätze benötigen. In dieser Arbeit wird also

untersucht, ob es möglich ist aus Eye Tracking Aufnahmen, ohne den Zusatz anderer Daten, Trace Links zu erzeugen.

4. Anforderungsanalyse

Ziel der Anforderungsanalyse ist es, möglichst genau zu beschreiben, welche Funktionen in der zu entwickelnden Anwendung umgesetzt werden sollen. Dies dient im Verlauf der Arbeit als Leitfaden, um zum gewünschten Ergebnis zu kommen.

Dieses Kapitel ist aufgeteilt in die funktionalen Anforderungen, bezeichnet mit „R“, und die nichtfunktionalen Anforderungen, die mit „NR“ gekennzeichnet werden. Alle Anforderungen werden anhand der Vorlage der SOPHISTen formuliert, sodass Missverständnisse durch ungenaue Sprache vermieden werden und die Anforderungsanalyse strukturiert durchgeführt werden kann [21].

4.1. Funktionale Anforderungen

Funktionale Anforderungen beschreiben die spezifischen Aktionen, Funktionen oder Verhaltensweisen, die ein System, eine Software oder ein Produkt erfüllen oder unterstützen soll. Sie definieren, was das System tun muss, um die Bedürfnisse der Benutzer oder Kunden zu erfüllen.

R01 *Die Anwendung muss dem Benutzer die Möglichkeit bieten, einzelne Eingabedateien in die Anwendung zu laden.*

Die Trace Links sollen anhand von bereits aufgezeichneten Eye Tracking Daten erzeugt werden. Dies erfordert die Eingabe dieser Eye Tracking Daten über einen Dateiimport aus einem vom Nutzer zu wählenden Speicherort.

R02 *Nachdem der Benutzer eine Datei zum Import ausgewählt hat, muss die Anwendung die Struktur der Eingabedatei auf Richtigkeit prüfen.*

Eye Tracking Daten liegen in einer vordefinierten Struktur vor. Damit die Anwendung alle benötigten Daten automatisiert finden kann, muss die Eingabedatei auf die vorgegebene Struktur überprüft werden.

R03 *Die Anwendung muss dem Nutzer die Möglichkeit bieten, Heuristiken für das Filtern der Trace Links auszuwählen.*

Für das Filtern der potentiellen Links soll die Anwendung Heuristiken zu Verfügung stellen, die vom Nutzer ausgewählt werden können. Dies soll es dem Nutzer ermöglichen die erzeugten Links für die weitere Nutzung zu optimieren.

R04 *Die Anwendung muss dem Nutzer die Möglichkeit bieten, die ausgewählten Heuristiken zu konfigurieren.*

Die Heuristiken, die der Nutzer ausgewählt hat, sollen konfigurierbar sein, damit es möglich ist, die Heuristiken an die jeweilige Aufnahme anzupassen.

R05 *Nachdem der Nutzer die Heuristiken konfiguriert hat, muss die Anwendung die Möglichkeit bieten, die Berechnung der Links zu starten.*

Damit die Oberfläche möglichst benutzerfreundlich ist, soll die Anwendung eine Schaltfläche beinhalten, die das Starten der Berechnung ermöglicht.

R06 *Während der Berechnung der Trace Links sollte die Anwendung dem Nutzer die Möglichkeit bieten, den aktuellen Fortschritt nachzuvollziehen.*

Je nach Leistung des verwendeten Rechners, kann die Verarbeitungszeit der Anwendung stark variieren, sodass auch mit Wartezeiten von mehreren Sekunden zu rechnen ist. Damit der Nutzer einen Absturz der Anwendung ausschließen kann, sollte die Anwendung Rückmeldung über den aktuellen Fortschritt geben.

R07 *Nachdem die Anwendung die Links berechnet hat, muss die Anwendung die erzeugten Links übersichtlich darstellen.*

Diese Anforderung dient der Nutzerfreundlichkeit der Anwendung. Durch eine übersichtliche Darstellung der Links kann der Nutzer schnell feststellen, ob das Ergebnis zufriedenstellend ist, oder ob weitere Konfigurationen an der Filterung und an den Heuristiken vorgenommen werden müssen.

R8 *Nach dem Berechnen und Anzeigen der Links soll Anwendung dem Nutzer die Möglichkeit bieten, die Konfiguration der Heuristiken zu verändern.*

Sind die Heuristiken nicht korrekt konfiguriert, kann es vorkommen, dass die erzeugten Links nicht den Vorstellungen des Nutzers entsprechen. Um ein einfaches Rekonfigurieren zu ermöglichen, soll eine Schaltfläche zur Verfügung gestellt werden, um zurück zur Konfiguration zu springen.

R9 *Nachdem die gewünschten Trace Links erstellt wurden, sollte die Anwendung die Möglichkeit bieten die Ergebnisse als JSON-Datei zu exportieren.*

JSON-Dateien können zum einen große Datenmengen bewältigen, zum anderen bieten sie auch die Möglichkeit Inhalte unkompliziert in andere Programme zu übertragen. Außerdem lassen sich in einer JSON-Datei verschachtelte Datenstrukturen, wie die hier erzeugten Links effizient darstellen. Daher sollte es möglich sein, die Ergebnisse als JSON-Datei zu exportieren.

4.2. Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen beschreiben Qualitätsmerkmale eines Systems, einer Software oder eines Produkts, die über die reine Funktionalität hinausgehen. Sie betreffen Aspekte wie Performance, Sicherheit, Benutzerfreundlichkeit oder Skalierbarkeit und definieren Anforderungen an die Qualität und das Verhalten des Systems.

NR01 *Die Anwendung sollte wartbar und skalierbar sein.*

Nach Abschluss der Arbeit soll es möglich sein, dass weitere Heuristiken eingepflegt werden können und die Anwendung einfach wartbar ist.

NR02 *Nachdem die Heuristiken ausgewählt und angewandt wurden, sollte die Anwendung alle Trace Links ausgeben.*

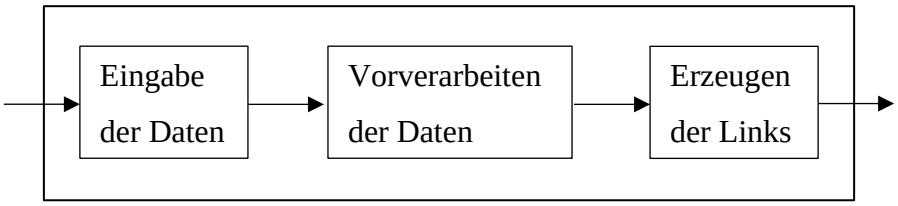
Die Trace Links lassen sich manuell nur schwer erzeugen. Daher muss der Nutzer davon ausgehen können, dass die Anwendung vollständig und zuverlässig arbeitet.

NR03 *Die Anwendung muss problemlos auf einem System mit Windows 10 zu installieren sein.*

Windows 10 war im Januar 2023 mit Abstand das am weitesten verbreitete Betriebssystem auf Desktop Systemen [22]. Daher soll die Anwendung auf allen Systemen mit Windows 10 ausführbar sein.

5. Konzept

Nachdem zuvor die Anforderungen für die Anwendung detailliert dargestellt wurden, wird in diesem Kapitel auf Basis dessen ein Konzept entwickelt. Die Anwendung besteht im Wesentlichen aus 3 Bereichen, auf die in den folgenden Unterkapiteln eingegangen wird.



5.1. Eingabe der Daten

Zu Beginn muss der Nutzer die Möglichkeit haben, Eingabedaten in die Anwendung zu importieren. Die Anforderungen R01 und R02 lassen sich zu diesem Abschnitt zuordnen. Dazu soll sich zunächst ein separates Fenster öffnen, welches es erlaubt, einen Speicherort auszuwählen, an dem die zu öffnenden Daten liegen. Voraussetzung hierfür ist, dass die Datei im CSV- oder TSV-Format (im Folgenden nur CSV) vorliegen.
Um sicherzustellen, dass die Datei korrekt verarbeitet werden kann, muss anschließend überprüft werden, ob alle nötigen Spalten in der CSV-Datei vorhanden sind. Für das Verarbeiten der Daten sollen stets folgende Spalten vorhanden sein:

Recording Timestamp:

- Diese Spalte dient zur Berechnung der Dauer einer Betrachtung

Project name, Participant name, Recording name, Recording date, Recording duration:

- Dies sind Metadaten, die genutzt werden können, Aufnahmen voneinander zu unterscheiden

Event, Event value, Gaze point X, Gaze point Y

- Diese Spalten beinhalten Informationen über den aktuellen Blickpunkt

Validity left, Validity right:

- Mit diesen Spalten kann validiert werden, dass die Daten in der jeweiligen Zeile korrekt aufgenommen wurden.

Eye movement type, Gaze event duration:

- Diese Spalten ordnen die Blickpunkte in Fixationen und Sakkaden ein

AOI hit:

- Für jede definierte AOI ist hier eine Spalte aufgelistet, die Informationen über den aktuellen Status der jeweiligen AOI beinhaltet.

5.2. Filterung der Daten

Eye Tracking Daten bestehen in der Regel aus einer Reihe für diese Anwendung nicht notwendigen Daten. So werden z.B. Kalibrierungseigenschaften, Datum und Uhrzeit der Aufnahme und Bezeichnung des Eye Trackers sowie weitere Eckdaten der Aufnahme gespeichert. Diese nehmen jeweils eine eigene Spalte in der CSV-Datei ein und müssen, damit das weitere Verarbeiten der Daten nicht zu lange dauert, zuvor zusammengefasst oder entfernt werden. In Abbildung 3 ist ein Ausschnitt einer Eye Tracking Datei zu sehen, nachdem diese aus Tobii Pro Lab exportiert wurde,.

Abbildung 3 - Auszug einer Eye Tracking Datei

Die in der Abbildung rot markierten Spalten enthalten Daten, die für die Berechnung der Links unerheblich sind. Gelb markiert sind die Metadaten, die nur zum Zuordnen der Aufnahme benötigt werden. Blau und Grün markiert sind die tatsächlich genutzten Daten. Dazu gehören zum einen die Augenpositionen und Timestamps (grün), zum anderen die definierten AOIs (blau). In der Abbildung ist zu erkennen, dass ein wesentlicher Teil der Daten nicht benötigt wird und daher vor dem Berechnen der Links gefiltert werden sollte.

5.2.1. Vorverarbeitung der Daten

Nachdem eine CSV-Datei eingegeben wurde, müssen zunächst alle nicht benötigten Zeilen und Spalten gefiltert werden, sodass sich die Eingabedatei für die anschließenden Berechnungsschritte möglichst klein ist. Der Start der Aufzeichnung besteht in der Regel aus der Kalibrierung des Eye Trackers. Diese Daten sind für die Anwendung unerheblich und können daher vernachlässigt werden. Auch Zeilen in denen der „Eye movement type“ als „Eyes-NotFound“ deklariert ist, können ignoriert werden. Auch diese beinhalten keine nützlichen Informationen für die Berechnung der Links, da hier die Aufnahme beider Augen als ungültig markiert ist und somit kein Blickpunkt bestimmt werden konnte.

Die CSV-Datei beinhaltet außerdem eine ganze Reihe an Metadaten der Aufnahme. Diese können in einer geeigneten Datenstruktur gespeichert werden, um diese in der Nutzeroberfläche anzeigen zu können. Da diese Daten aber einen erheblichen Teil der gesamten CSV-Datei ausmachen, werden diese Spalten nach der separaten Speicherung für die weitere Berechnung der Trace Links nicht mehr berücksichtigt.

Für die restlichen Daten, die für die Berechnung relevant sind, wird eine eigene Datenstruktur angelegt, die im folgenden Abschnitt beschrieben wird.

5.2.2. Datenstrukturen

Die Eye Tracking Daten sollen nach der Eingabe in eine Datenstruktur überführt werden, die einen effizienten Zugriff auf die wesentlichen Daten erlaubt. Diese besteht zum einen aus einer Liste von AOI-Objekten, in denen eine einzigartige ID, der Name, sowie eine Liste aller Fixationen und Sakkaden

und einer Liste aller Betrachtungen der AOI, die durch ein Tupel aus Start- und Endzeitpunkt dargestellt werden, gespeichert wird. Die Fixationen sind ebenfalls in einer eigenen Klasse definiert, die Start- und Endzeitpunkt und Typ der Blickbewegung speichert (Fixation, Sakkade oder nicht klassifiziert). Zum andere enthält die Datenstruktur ein Objekt für die Metadaten der Aufnahme. Hier werden die Namen des Projekts, der Aufnahme und des Teilnehmers gespeichert, sowie Eckdaten der Aufnahme, wie Datum, Dauer und Start- und Endzeitpunkt.

Für das Übertragen der Links an die Oberfläche sollen auch die Links in einer Datenstruktur abgelegt werden, die aus den verknüpften AOIs und dem berechneten Score besteht. Die folgende Abbildung 2 zeigt ein Klassendiagramm der Datenstruktur zum Speichern der Eye Tracking Daten und der Trace Links.

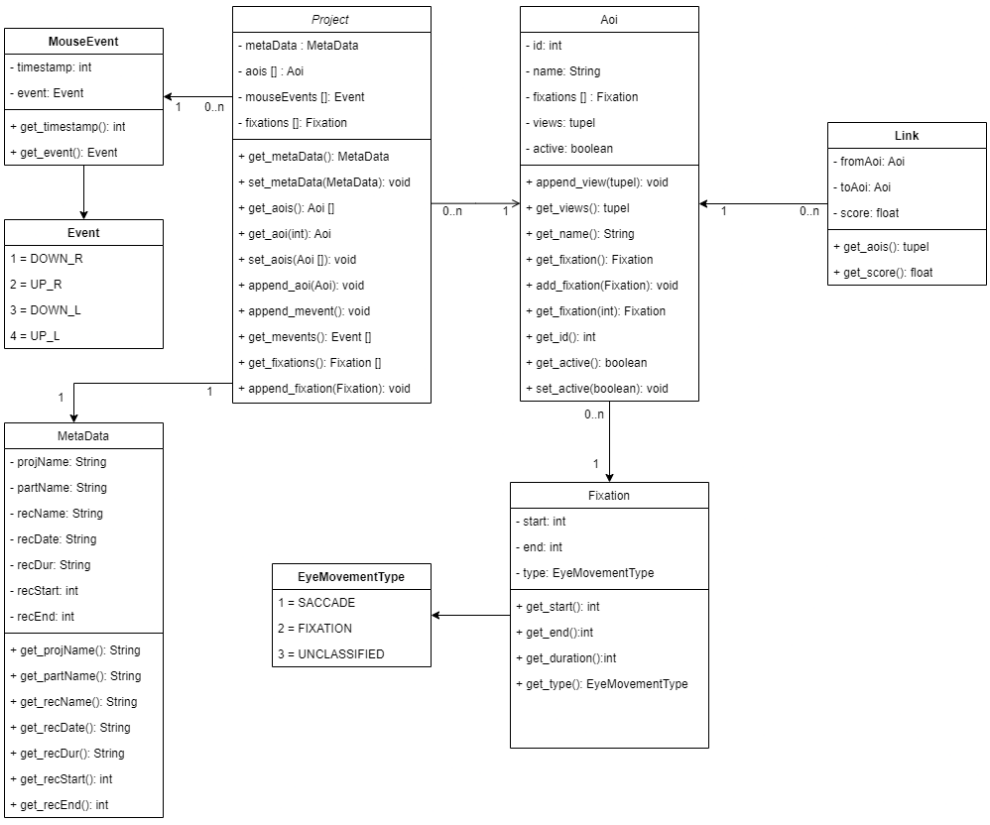


Abbildung 4- Datenstruktur zum Speichern der Eye Tracking Daten

5.3. Erzeugen der Trace Links

Das Erzeugen der Links startet mit einer n x n Matrix, wobei n die Anzahl der AOIs ist. Die Matrix soll nach der Berechnung der Links die Scores der jeweiligen AOIs enthalten. So entspricht beispielsweise ein Score von 0,6 an der Stelle [3, 5] einem Link mit dem Score 0,6 zwischen den AOIs mit der IDs 3 und 5. Jeder mögliche Link erhält während der Berechnung einen Score zwischen 0 und 1. Eine 0 bedeutet hier, dass zwischen den beiden AOIs keine Verbindung besteht, eine 1 hingegen, dass der Link sehr relevant ist. Im Folgenden wird erläutert, nach welchen Heuristiken die Scores berechnet werden und welche Schnittstelle zur Verfügung gestellt wird, um weitere Heuristiken in die Anwendung einzupflegen.

5.3.1. Heuristiken

Die Anwendung muss für die Filterung der Links einige Heuristiken bereitstellen. Diese sollen über die Nutzeroberfläche ausgewählt und konfiguriert werden. Heuristiken werden durch Plugins in die Anwendung eingebunden. Jedes Plugin generiert separat einen Score für alle möglichen Links zwischen allen AOIs. Nach dem Ausführen aller Plugins wird der gewichtete Mittelwert aller Scores für jeden Link berechnet. Die Gewichtung jeder Heuristik kann beim Konfigurieren der Plugins angepasst werden. Dies ermöglicht es dem Nutzer Heuristiken, die besser auf die jeweilige Aufnahme passen, vermehrt in die Berechnung einfließen zu lassen. Auch Heuristiken, die erst im Nachhinein entwickelt werden, sollen angezeigt und konfiguriert werden können. Im Laufe der Arbeit werden bereits einige grundlegende Plugins zum Erzeugen von Links entwickelt:

Das erste Plugin beinhaltet eine Heuristik, die Links anhand des zeitlichen Abstands zwischen dem Betrachten der AOIs erzeugt. Eine solche Heuristik auf die Eye Tracking Daten anzuwenden, kann sinnvoll sein, da beim Entwickeln einer Software, oder beim Aufstellen von Requirements in der Regel verknüpfte Elemente zeitnah zueinander betrachtet werden. Dieses Plugin vergibt also höhere Scores an Links zwischen AOIs, deren Betrachtungen zeitlich näher zusammen liegen. Hierzu werden für die beiden betrachteten AOIs alle zeitlichen Abstände ermittelt und gespeichert. Anschließend wird

der Durchschnitt ermittelt. Dadurch wird sichergestellt, dass sehr stark abweichende Werte korrigiert werden. Abschließend wird aus den durchschnittlichen Werten ein Score berechnet. Hierfür soll keine lineare Funktion verwendet werden, sondern eine Polynomfuntktion. Dies erzeugt eine andere Verteilung der Scores, sodass der Score mit steigendem zeitlichem Abstand zunächst nur leicht sinkt und immer schneller auf 0 sinkt, je mehr sich der tatsächliche Abstand dem konfigurierbaren Abstand nähert. Den maximalen zeitlichen Durchschnittsabstand konfigurierbar zu machen, bietet den Vorteil, dass so die Scores für unterschiedliche Aufnahmen besser anpassbar sind. Aufnahmen mit schnell wechselnden Tasks und Aufgaben sollten beispielsweise bessere Scores erhalten, wenn der Wert kleiner gewählt wird, als bei Aufnahmen mit langen Tasks.

Das zweite Plugin erzeugt Links basierend auf den direkt auf einander folgenden Betrachtungen der AOIs. Wird häufig zwischen zwei AOIs hin- und hergeschaut, so kann davon ausgegangen werden, dass die beiden Elemente der AOIs mit einander verknüpft sind. Beim Erstellen von Requirements und Konzepten wird beispielsweise häufig zwischen dem jeweiligen zu bearbeitenden Dokument und eventuellen Kundenwünschen oder anderen Anforderungen gewechselt. Das Gleiche Prinzip lässt sich auch auf die Entwicklung von Software anwenden. Auch wir werden verknüpfte Teile des Quellcodes meist nacheinander betrachtet. Eine solche Heuristik kann also ebenfalls sinnvoll sein. Die Scores für potentielle Links werden hier berechnet, indem für jede mögliche Kombination an AOIs die aufeinanderfolgenden Betrachtungen gezählt werden. Das Ergebnis wird anschließend mit Division durch den höchsten ermittelten Wert normiert und so ein Score zwischen 0 und 1 erzeugt.

Das dritte im Rahmen dieser Arbeit entwickelte Plugin betrachtet ebenfalls den Abstand der Betrachtungen von AOIs in diesem Fall wird allerdings nicht der zeitliche Abstand betrachtet, sondern die Anzahl der zwischenzeitlich betrachteten AOIs. Dies könnte ebenfalls eine valide Möglichkeit sein, AOIs mit einander zu verknüpfen, da es auch Arbeitsabläufe gibt, in denen zuerst andere AOIs betrachtet werden und dennoch ein Link zwischen den AOIs besteht. Das Plugin soll für jede Kombination von AOIs die

dazwischenliegenden Betrachtungen ermitteln und diese speichern. Anschließend wird der Durchschnitt ermittelt, um einen repräsentativen Wert zu erhalten. Minimalwerte können an dieser Stelle irreführend sein, da einzelne Blickwechsel zwischen AOIs den Score so verfälschen. Abschließend wird das Ergebnis normiert, um einen Score zwischen 0 und 1 zu erhalten.
Das letzte Plugin ist nicht direkt zum Erzeugen von Links gedacht. Alleingestellt hebt es lediglich die am meisten betrachteten AOIs hervor. Hierfür wird für jede AOI ermittelt, wie lange diese insgesamt betrachtet wurde. Dieser Wert wird anschließend ebenfalls auf einen Wert zwischen 0 und 1 normiert. Der Zweck dieses Plugins besteht darin, die AOIs hervorzuheben, die besonders wichtig für das jeweilige Projekt sind. Wichtige Elemente werden meist häufiger und/oder länger betrachtet, weshalb sich hier ein größerer Wert ergibt und sich Links dieser AOIs insgesamt besser bewerten lassen.

5.3.2. Erweiterungen

Die in 5.3.1 beschriebenen Heuristiken sind als Plugin zu implementieren. Damit wird sichergestellt, dass die nichtfunktionale Anforderung NR01 erfüllt ist. Eine Plugin-Struktur ermöglicht es im Nachhinein weitere Heuristiken in die Anwendung einzupflegen. Dazu wird eine geeignete Schnittstelle entwickelt, die zur Entwicklung solcher Plugins dient.
Diese Schnittstelle besteht zum einen aus einer abstrakten Klasse, die die Struktur für weitere Plugins vorgibt. Die Klasse besitzt eine Methode zur Initialisierung des Plugins. Hier werden z.B. UI-Elemente wie Inputs oder Dropdowns deklariert, die zur Konfiguration der Heuristiken verwendet werden können, aber auch Konstanten und andere benötigte Variablen können deklariert und initialisiert werden. Des Weiteren gibt es eine Methode, die die Heuristik anwendet. Sie dient als Einstiegspunkt in das Plugin und startet die Berechnung der Links.
Zum anderen bietet auch die Nutzeroberfläche der Anwendung die Möglichkeit, die in den Plugins definierten UI-Elemente anzuzeigen. Dazu wird ebenfalls eine geeignete Datenstruktur bereitgestellt, die die Definition von Dropdowns und Zahlen-/Text-Inputs ermöglicht.

6. Implementierung

In diesem Kapitel wird auf die Implementierung des Prototyps zum Filtern der Trace Links beschrieben. Bei dem Prototyp handelt es sich um eine Webanwendung, die in der Programmiersprache Python (Version 3.10.8) entwickelt wurde. Im Verzeichnis der Anwendung ist ebenfalls ein Dockerfile, sowie eine docker-compose.yml Datei enthalten. Diese ermöglichen es dem Nutzer, die Anwendung auf jedem Gerät, das Docker unterstützt, laufen zu lassen, und erfüllt somit die Anforderung NR03. Docker ist ein Tool, das es erlaubt, konsistente Laufzeitumgebungen zu erzeugen, die isoliert vom restlichen System ausgeführt wird [23].
Im folgenden Abschnitt 6.1. werden die verwendeten Bibliotheken beschrieben. Abschnitt 6.2. behandelt die Architektur der Anwendung.

6.1. Verwendete Bibliotheken

Für die Implementierung der Anwendung wurde vor allem das Webframework Flask verwendet. Flask ist eines von vielen Webframeworks für Python. Hier wurde es vor allem auf Grund des integrierten Development Servers, der es erlaubt in kürzester Zeit Änderungen an der Anwendung im Browser zu testen, und der Jinja Template Engine, die eine einfache Möglichkeit bietet, die verschiedenen Seiten der Anwendung über ein Template einheitlich zu gestalten, gewählt. Darüber hinaus wird für die Berechnung innerhalb der Plugins die Bibliothek numpy verwendet. Diese ermöglicht unter anderem das effiziente Verarbeiten von Matrizen und Arrays in Python und bietet somit viele Tools, die für die Berechnungen genutzt werden können [24].

6.2. Architektur der Anwendung

Im Folgenden wird die Architektur anhand eines groben Programmablaufs beschrieben und auf einige Implementierungsdetails eingegangen.
In main.py wird nach dem Starten der Anwendung eine Instanz von Flask gestartet und konfiguriert. Die Konfiguration beinhaltet das Speichern von relativen Pfaden für die Jinja-Templates, die CSS-Files und hochgeladene

Dateien. Außerdem wird hier der Ordner für hochgeladene Dateien geleert, falls noch Dateien aus einer vorherigen Nutzung vorhanden sind.

6.2.1. Views

Flask stellt ein eigenes Routing zur Verfügung. Damit können Routen erstellt werden, die Funktionen mit einer URL verknüpfen. Um auf diese Routen zugreifen zu können muss der Nutzer lediglich die Base-URL der Webanwendung gefolgt von dem jeweiligen Postfix aufrufen.
Es sind vier Routen implementiert. Die erste wird mit dem „/“ Postfix aufgerufen, sobald sich ein Nutzer mit der Webanwendung verbindet. Die Hauptfunktion der Route ist es, dem Nutzer zu ermöglichen eine TSV-Datei hochzuladen und somit die Anforderung R01 zu implementieren. Sobald die Eingabe geschehen ist, wird die Datei mit Funktionen des CSV-Handler, der in Abschnitt 6.2.2. näher beschrieben wird, auf die geforderte Struktur überprüft und die Metadaten der Aufnahme angezeigt. Mit einem Klick auf „Continue“ wird die nächste Route mit dem Postfix „/heutistics“ aufgerufen.
Die Hauptfunktion dieser Route ist die Konfiguration der Plugins und damit die Umsetzung der Anforderungen R03-R05. Hierfür werden zunächst die hochgeladenen Daten mit Funktionen des CSV-Handlers gefiltert und in der im Konzept (Abschnitt 5.2.2.) beschriebenen Datenstruktur gespeichert. Zusätzlich dazu werden mit dem Plugin-Processor, der in Abschnitt 6.2.3. näher beschrieben wird, alle Plugins aus dem Plugins-Ordner geladen. Anschließend wird das heuristicsettings.html Template gerendert um dem Nutzer das weitere Konfigurieren zu ermöglichen. Mit einem erneuten Klick auf „Continue“ wird die Konfiguration übernommen, die apply_heuristics Funktion der Plugins ausgeführt und die nächste Route aufgerufen.
Diese Route hat das Postfix „/results“ und dient dazu die Gesamtergebnisse anzuzeigen. Die Ergebnisse der einzelnen Plugins werden hierzu mit den vom Nutzer definierten Gewichtungen multipliziert und aufaddiert. Nach dem Normieren werden die Links in einen separate Datenstruktur überführt, die jeweils aus den beiden AOIs, die in dem Link verknüpft sind, und dem berechneten Score besteht. Das Übertragen in diese Datenstruktur vereinfacht

im Anschluss das Anzeigen der Links, da hier lediglich die Liste der Links iteriert und dargestellt werden muss.
Die letzte Route „/export“ ermöglicht den Datenexport als JSON-Datei und implementiert dadurch die Anforderung R09. Diese Datenstruktur wurde gewählt, weil das JSON-Dateiformat optimal dafür geeignet ist, verschachtelte Datenstrukturen, wie die hier erzeugten Links, abzubilden.

6.2.2. CSV-Handler

Der CSV-Handler stellt drei wesentliche Funktionen zum Verarbeiten der eingegebenen TSV- bzw. CSV- Dateien zur Verfügung. Die erste Funktion (check_valid) prüft, ob die Eingabedatei eine geeignete Struktur vorweist um anschließend von der Anwendung verarbeitet werden zu können und dient somit der Umsetzung von Anforderung R02. Die zweite Funktion (filter_data) filtert die Daten der Eingabedatei und speichert diese in einer separaten input.csv Datei ab. Das Filtern der Daten dient lediglich zum Verbessern der Performanz der Anwendung, da so ein Großteil der Daten aus der Eingabedatei entfällt. Die letzte wesentliche Funktion (import_data) wandelt die zuvor generierte input.csv in die im Abschnitt 5.2.2. beschriebene Datenstruktur um, sodass die Daten den Plugins stets in einer einheitlichen Struktur zur Verfügung stehen.

6.2.3. Plugin-Processor

Der Anwendung sollen im Nachhinein weitere Heuristiken hinzugefügt werden können (Anforderung NR01). Dies wird durch ein Plugin Modell implementiert. Der Plugin-Processor implementiert hierfür zwei Funktionen. Mit der ersten Funktion (load_plugins) werden alle Dateien im Plugins-Ordner nach Klassen durchsucht, die von der abstrakten Klasse HeuristicPlugin erben, importiert diese dynamisch und übergibt eine Liste mit allen importierten Klassen. Die zweite Funktion (exec_plugins) ruft im Anschluss von allen importierten Plugins die apply_heuristic Funktion auf und gibt eine Liste zurück, die die Ergebnisse aller Plugins enthält.

6.2.4. Plugins

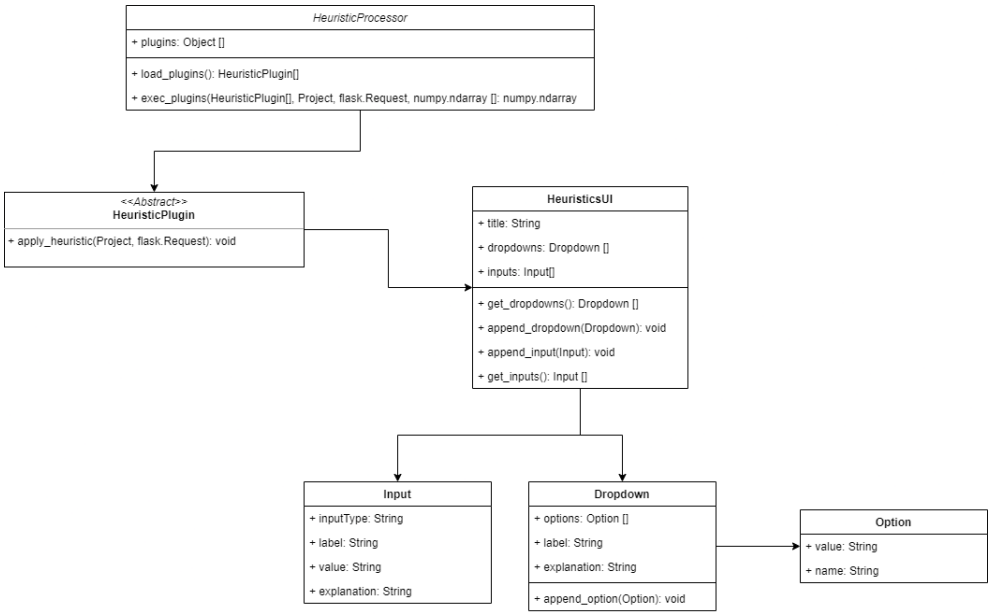


Abbildung 5- Klassendiagramm PluginProcessor

Das Generieren der Links wird nicht in der eigentlichen Anwendung, sondern in Plugins umgesetzt. Diese müssen von der Klasse HeuristicPlugin erben, um die benötigte Struktur zu gewährleisten. Die apply_heuristic Funktion stellt hierbei den Einstiegspunkt in die Berechnung der Links dar und muss ein ndarray von numpy zurückgeben, das eine Größe von n x n hat, wobei n hier die Anzahl der AOIs ist. Dieser Funktion werden sowohl die Eye Tracking Daten in Form der in Abschnitt 5.2.2. beschriebenen Datenstruktur als auch der HTTP POST Request der „/heuristicsettings“ Route übergeben. Aus dem POST-Request kann das Plugin die Eingaben aus der Konfigurationsseite laden. Die Abbildung 5 zeigt ein Klassendiagramm der im Prototyp verwendeten Plugin-Schnittstelle.

Um Eingabefelder für die Konfiguration anzulegen, gibt es eine HeuristicUI Klasse, von der im Konstruktor des Plugins eine Instanz erstellt werden kann. Hier können HTML-Inputs und -Dropdowns angelegt und angepasst werden. Auf den Inhalt der Inputs und Dropdowns kann mit dem jeweils generierten Namen zugegriffen werden. Dieser besteht stehts aus dem im Plugin festgelegten Namen, gefolgt von _input_ für ein Input oder _dropdown_ für ein Dropdown, gefolgt vom Index in der im Konstruktor erzeugten Liste von Inputs bzw. Dropdowns. Soll zum Beispiel das erste Dropdown vom Plugin mit

dem Namen „timeDistance“ aufgerufen werden, so muss im Request nach „timeDistance_dropdown_0“ gesucht werden.

7. Evaluation

In diesem Abschnitt werden zunächst die für die Entwicklung verwendeten Eye Tracking Daten vorgestellt. Anschließend wird das Vorgehen bei der Evaluation der Ergebnisse vorgestellt und die Durchführung beschrieben. Zum Schluss folgt eine Analyse der aufgetretenen Probleme mit einigen Lösungsansätzen.

7.1. Verwendete Eye Tracking Daten

Um das Problem im Requirements Traceability am besten zu adressieren, wurden die für die Entwicklung der Anwendung benötigten Eye Tracking Aufnahmen in einem Unternehmen für Software Entwicklung durchgeführt. Insgesamt wurden so zwei 40-50 Minuten lange Arbeitsabläufe von zwei unterschiedlichen Entwicklern mit dem Eye Tracker Tobii Pro X3-120 aufgezeichnet. Die gewählte Aufnahmedauer resultiert daraus, dass die Kalibrierung des Eye Trackers mit der Zeit ungenauer wird und die Daten nach längerer Zeit vermehrt unbrauchbar werden.

In der ersten Aufnahme wurde ein Teil eines Requirements in der Webanwendung des Unternehmens durchgeführt. Hier existierte bereits ein Filter für eine angezeigte Liste, der in diesem Task um eine weitere Filtermöglichkeit erweitert werden sollte. Der zweite Teil der Aufnahme besteht aus einem weiteten Task, in dem eine Funktion entwickelt werden soll, die veraltete Daten migrieren soll.

Die zweite Aufnahme hingegen zeigt das Beheben zweier Bugs in der Desktopanwendung des Unternehmens. Die Aufnahme folgt also einem anderen Arbeitsablauf im Vergleich zur ersten Aufnahme, da hier zunächst die bearbeiteten Bugs nachgestellt werden und anschließend bearbeitet werden, während das Nachstellen bei der ersten Aufnahme, da hier neue Features umgesetzt werden, komplett entfällt.

Anschließend wurden in beiden Aufnahmen manuell mittels des Programms Tobii Pro Lab die benötigten AOIs deklariert. Für diesen Schritt steht auch eine OCR-Anwendung zur Verfügung, die allerdings für Aufnahmen dieser

Länge enorm viel Zeit benötigt. Für eine der beiden Aufzeichnungen brauchte die OCR (auf einem Rechner mit i7-6700k, 16 GB RAM und GTX 1070) hochgerechnet etwa 40 Stunden, sodass hier auf das manuelle Deklarieren der AOIs zurückgegriffen wurde. Im direkten Vergleich werden so ebenfalls deutlich weniger AOIs eingezeichnet, da die OCR auch z.B. UI-Elemente der Entwicklungsumgebung als AOI deklariert, die für das Erstellen der Links unerheblich sind.

Nachdem alle AOIs eingetragen wurden, konnten die Aufnahmen einzeln aus Tobii Pro Lab exportiert werden, sodass die Daten in einer TSV-Datei vorliegen. Auf das Format dieser Dateien ist die entwickelte Anwendung angepasst, sodass diese direkt als Eingabedateien verwendet werden können.

7.2. Vorgehensweise

Für die Evaluation der erzeugten Links wurden während der Entwicklung des Prototyps für beide Aufnahmen manuell eine Reihe von potentiellen Links erstellt (Anhang A.1). Diese Links wurden von den jeweiligen Entwicklern bewertet und dienen als Grundlage für die Bewertung der von der Anwendung erzeugten Links. Um die Liste der Links möglichst nichtsuggestiv zu gestalten, werden zu den potentiell erwünschten Links weniger relevante bis hin zu irrelevanten Links gemischt. Die Listen haben eine Länge von etwa 30 Links, da die Aufnahmen zum Zeitpunkt der Bewertung der Links bereits einige Wochen in der Vergangenheit lagen und das Nachvollziehen und Ausfüllen der Listen nicht länger als eine halbe Stunde dauern soll, um ein Abflachen der Konzentration beim Ausfüllen zu verhindern. Für die Bewertung der Links wurde eine Skala von 0 bis 10 gewählt. Diese Skala wurde gewählt, da so eine ausreichende Granularität entsteht, um die Links treffend zu bewerten. Feinere Skalen hätten keinen Mehrwert geboten, da hier in der Regel nicht mehr zwischen benachbarten Werten unterschieden werden kann.

Die Scores der von der Anwendung erzeugten Links werden anschließend mit den Scores verglichen, die die Entwickler vergeben haben. Hierzu werden zunächst die Bewertungen der Entwickler normiert, um die Scores und die Bewertungen vergleichen zu können. Anschließend wird für jeden Link die Differenz ermittelt. Für eine Bewertung der Ergebnisse wird anschießend der

Mittelwert und die Standardabweichung der Differenzen berechnet. Werte gegen 0 zeigen dabei eine große Übereinstimmung zwischen dem erzeugten Score und der Bewertung der Entwickler.

7.3. Evaluation der Scores

Beim Vergleichen der Scores, die durch den Prototyp berechnet wurden, mit den Bewertungen der Entwickler zeigt sich, dass der Prototyp zwar eine Reihe an Links erzeugt, jedoch nur 4 der 62 Links erkennt, die von den Entwicklern bewertet wurden. Die erkannten Links weichen mit einem Mittel von 0,1035 und einer Standardabweichung der Differenz von 0,3 ebenfalls sehr stark von den Bewertungen der Entwickler ab. Aufgrund dessen wird an Stelle einer Evaluation der erzeugten Scores und Links eine Problemanalyse des Entwickelten Prototyps durchgeführt, die im folgenden Abschnitt erläutert wird.

7.4. Problemanalyse und Lösungsansätze

Die ungenügenden Ergebnisse des Prototyps basieren auf einer Reihe von Problemen in der Zielsetzung, im Konzept, der Umsetzung des Prototyps und der Vorgehensweise während der Evaluation.

Zu Beginn der Arbeit wurde das Ziel gesetzt, mit einem Prototyp zu zeigen, dass es möglich ist nur anhand von Eye Tracking Daten von unterschiedlichen Stakeholdern des Software Entwicklungsprozesses sinnvolle Trace Links zu generieren. Eye Tracking Daten beinhalten im Kern lediglich die Informationen, wann der Nutzer wo hinschaut. Aus der Blickabfolge sollte also erkannt werden, welche Elemente mit einander verknüpft sind. Dies lässt jedoch einige wichtige Daten außer Acht.

Zum einen wurde nicht bedacht, dass unterschiedliche Stakeholder unterschiedliche Prozessabläufe haben, und so nicht mittels einfacher Regelsätze passende Trace Links aus jeder Eye Tracking Aufnahme generiert werden können. Beim direkten Vergleich der beiden im Verlauf der Arbeit aufgezeichneten Aufnahmen wird ersichtlich, dass sogar Entwickler im selben Unternehmen sehr unterschiedliche Arbeitsabläufe haben. Während der Entwickler, der an einem Task für die Desktop-Anwendung arbeitet, beispielsweise eine beträchtliche Zeit warten muss, bis der Quellcode kompiliert

wurde und die Ergebnisse seiner Arbeit getestet werden können, enthält der Arbeitsablauf des Entwicklers, der einen Task für die Webanwendung bearbeitet, nahezu keine Wartezeiten. Das führt dazu, dass wesentliche Teile des Konzepts nicht auf beide, oder sogar weitere Eye Tracking Aufnahmen, anzuwenden sind.

Zum anderen beinhalten die Eye Tracking Daten keine Informationen über die betrachteten Elemente. Die Information, um was für eine Art Element es sich handelt, ist jedoch in vielen Fällen sehr entscheidend. Ein Requirement wird beispielsweise im Workflow wesentlich anders betrachtet, als Tasks, Quellcode oder andere Dokumentationen. Könnten diese Informationen berücksichtigt werden, könnten deutlich komplexere Heuristiken definiert werden, wie beispielsweise die Regelsätze in der Arbeit von Spanoudakis et al. [1]. Hier wurde zwar ein Ansatz mit Machine Learning gewählt, dem Modell standen jedoch auch Informationen zur Verfügung, um welche Art von Element es sich handelt. So wurden hier Regelsätze erstellt, die anhängig von diesen wesentlichen Informationen Links erstellen können. Während der Evaluation der Arbeit von Spanoudakis et al. wurde unter anderem festgestellt, dass dieses Modell anderen Tools zur Erzeugung von Trace Links in den Precision- und Recall-Werten deutlich überlegen war. Dies kann unter anderem durch dir zuvor beschriebenen Heuristiken hervorgerufen werden.

Auch die Evaluation dieser Arbeit weist einige Probleme auf. In einem Großteil der verwandten Arbeiten wurden Tools zum Erzeugen von Trace Links mit Precision- und Recall-Werten bewertet. Um diese für diesen Prototyp berechnen zu können, hätte für beide Aufnahmen eine Auflistung aller Links erstellt werden müssen, um ermitteln zu können, wie groß der Anteil der korrekt erzeugten Links und wie groß der Anteil der falschen Links ist. Basierend auf den Werten der Evaluation, lässt sich jedoch vermuten, dass der Prototyp auch mit einer solchen Evaluation wesentlich schlechter abgeschnitten hätte, als bisherige Ansätze. Der zeitliche Aufwand diese Listen zu erstellen, wäre jedoch den Entwicklern nicht zumutbar gewesen, da dies mehrere Stunden dauern kann.

Und auch die in der Arbeit verwendeten Listen zur Bewertung der Links weisen Probleme auf. Die Listen, die den Entwicklern vorgelegt wurden, sind

parallel zum Entwickeln des Prototyps entstanden. Zu diesem Zeitpunkt gab es noch keine Ausgabedaten des Prototyps. Eine sinnvollere Variante der Evaluation könnte sein, die Entwickler nicht Listen mit vorab erstellten Links bewerten zu lassen, sondern ihnen die Ausgabe des Prototyps vorzulegen und die erzeugten Links bewerten zu lassen. Zusätzlich dazu müssten die Entwickler noch die Möglichkeit haben, Links, die nicht erzeugt wurden, aber dennoch von Bedeutung sind, aufzulisten und ebenfalls zu bewerten. Für diese Variante der Evaluation reicht jedoch der zeitliche Rahmen dieser Arbeit nicht aus, weshalb dieser Ansatz nicht verfolgt wurde.

Die hier aufgetretenen Probleme wurden bereits von Ahrens [25] in einer Arbeit hervorgehoben. Eye Tracking für das Erzeugen von Trace Links zu nutzen, erzeuge eine Vielzahl an false positive Links, also Links, die zwar berechnet wurden, aber keinen semantischen Mehrwert bieten.

8. Fazit und Ausblick

In diesem Abschnitt werden die Ergebnisse dieser Arbeit zusammengefasst und abschließend bewertet. Den Abschluss bildet ein Ausblick über mögliche weitere Arbeiten.

8.1. Fazit

In dieser Arbeit wurde ein Prototyp entwickelt, um zu untersuchen, ob es möglich ist Trace Links aus Eye Tracking Daten zu erzeugen. Diese Links ermöglichen es Stakeholdern in Softwareprojekten Arbeitsabläufe effizienter zu gestalten, indem Zusammenhänge schneller zu erkennen sind. Anders als bei anderen Arbeiten, in denen das Traceability Problem mit einem Eye Tracking Ansatz behoben werden soll, sollte dieser Prototyp ausschließlich basierend auf Eye Tracking Daten Links erzeugen, ohne Typisierungen von Dokumenten und andren Artefakten zu nutzen.

Für die Umsetzung wurde zunächst eine Anforderungsanalyse durchgeführt und anschließend ein Konzept entwickelt, das den Prototyp beschreibt. Der Prototyp bietet die Möglichkeit, Eye Tracking Daten über eine Auswahl von Heuristiken zu analysieren und Verbindungen zwischen den AOIs herzustellen. Darüber hinaus bildet er eine Grundlage, die es ermöglicht, in weiteren Arbeiten andere Heuristiken in die Software zu integrieren.

Die Evaluation mit bewerteten Eye Tracking Daten hat gezeigt, dass die generierten Links des Prototyps vergleichsweise stark von den Bewerteten Daten abweichen. Daraufhin wurde eine Problemanalyse durchgeführt, in der problematische Elemente der Anwendung identifiziert werden konnten. Eine Problemquelle sind die eingegebenen Daten. Da diese keine Informationen über den Typ der jeweiligen Dokumente bzw. der anderen Artefakte beinhalten, werden auf alle Elemente die gleichen Heuristiken angewendet. Spanoudakis et al. [1] zeigen in ihrer Arbeit jedoch, dass der Typ der Elemente durchaus ausschlaggebend sein kann. Ein weiteres Problem ist, dass der Prototyp für unterschiedliche Stakeholder ausgelegt sein sollte. Aus den Aufnahmen, die während der Arbeit angefertigt wurden, geht bereits hervor, dass selbst

Entwickler aus demselben Entwicklungsteam, gravierende Unterschiede in ihren Arbeitsabläufen aufweisen. Für eine Verallgemeinerung der Abläufe müssten deutlich mehr Daten erfasst und ausgewertet werden. Die Evaluation und Problemanalyse zeigen also, dass weitere Optimierungen nötig sind, verwandte Arbeiten legen jedoch nahe, dass es möglich sein kann, Trace Links basierend auf Eye Tracking Daten zu generieren.

8.2. Ausblick

Durch das Hinzufügen weitere Heuristiken kann die Funktionalität des Prototyps verbessert werden. Für diesen Schritt kann es sinnvoll sein, in weiteren Arbeiten Eye Tracking Daten von Entwicklern zu analysieren, um Muster in den Daten zu ermitteln, die hilfreich beim Entwickeln von Heuristiken für den hier entwickelten Prototyp sein können. Das Eclipse Plugin iTrace, das in der Arbeit von Shaffer et al. [16] vorgestellt wird, kann hierfür als Einstieg dienen, da das Tool neben der Aufzeichnung der Blickdaten die Code-Artefakte auch typisiert.

Eine weitere mögliche Verbesserung wäre es, neben den Eye Tracking Daten auch eine Art Typisierung der AOIs in den Prototyp einzugeben. Dies würde es erlauben, einzelne Typen von Elementen gesondert zu behandeln und Heurisiken zu entwickeln, die mehr Daten verarbeiten, als lediglich die Blickdaten der Nutzer.

Anhang

A.1 Linktabellen für Evaluation

Tabelle 1 – Bewertung der Links aus Aufnahme 1

Elemente des Links	0	1	2	3	4	5	6	7	8	9	10
Bug 19422: V43 VEM/INBOX Aktualisieren Button Datei - InboxModelView.cs											x
Datei - InboxModelView.cs TB4 VEM Inbox									x		
Bug 19422 Branch #19422-change-togglebutton-to-normal-button									x		
Datei – InboxModelView.cs Branch #19422-change-togglebutton-to-normal-button									x		
Bug 19422 TB4 Patientenstamm			x								
Bug 19422 Pull Request C43.46.2											x
Bug 19422 Datei – AppointmentItemActions.cs	x										
Bug 19422 Branch #19422-merge-v43-v44											x
Bug 19422 Pull Request C44.32.2											x
Bug 19422 Bug 19530			x								
Bug 19422 Brach #19422-merge-v44-v45											x
Bug 19422 Pull Request C45.44.25											x
Bug 19422 Branch #19422-merge-v45-main											x
Bug 19422 Datei - TbAssemblyInfo.cs											x

Bug 19428 V43 Menge/Frequenz in BSZ Rückseite ändern TB4 VEM Bestellungen								x			
Bug 19428 V43 Menge/Frequenz in BSZ Rückseite ändern TB4 BSZ Rückseite								x			
Bug 19428 TB4 Patientenstamm			x								
TB4 Patientenstamm TB4 VEM BSZ Summenansicht			x								
Bug 19530 INOR-Browser hängt sich auf Datei - EventHandler.cs	x										
Bug 19530 Branch #19422-merge-v44-v45		x									
Bug 19530 Datei - ShowAlertEventArgs.cs	x										
Bug 19530 TB4 VEM Bestellungen	x										
Bug 19530 TB4 BSZ Rückseite	x										
Bug 19530 TB4 INOR-Browser								x			
Bug 19571 Sperrzeitserie Text wird nicht in Folgetermine übernommen Datei – ShowAlertEventArgs.cs		x									
Bug 19571 Datei – AppointmentItemActions.cs	x										
Bug 19571 Datei – RecurringAppointmentDialogViewModel.cs				x							
Bug 19571 Datei – TbAssemblyInfo.cs								x			
Bug 19571 TB4 VEM Inbox	x										
Datei – AppointmentItemActions.cs Datei – RecurringAppointmentDialogViewModel.cs	x										

Tabelle 2 – Bewertung der Links aus Aufnahme 2

Elemente des Links	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Task 19317 neuer Filter „nur zu vidie-rende BSZ anzeigen“ Datei – EvoFilter.vue											x
Task 19317 Datei – EvoEditMode.vue								x			
Task 19317 Datei – EvoMigrationMode.vue								x			
Task 19317 Datei – EvoWorkspace.js								x			
Datei – EvoEditMode.vue Datei – MigrationEntry.vue								x			
Task 19317 TBWA Evo Filter											x
Task 19317 TBWA EVO Bestellzeilen Übersicht								x			
TBWA EVO Bearbeitungsmodus Datei – EvoMigrationMode.vue								x			
Task 19317 TBWA EVO Bearbeitungsmodus								x			
TBWA EVO Filter Datei - EvoFilter.vue											x
TBWA EVO Filter Datei – OrderLineHeader.vue									x		
TBWA EVO Filter Datei - EvoEditMode.vue								x			
Commit #19317 Datei - EvoFilter.vue										x	
Commit #19317 Datei – EvoMigrationMode.vue										x	
Commit #19317 Datei - EvoWorkspace.js										x	
Task 19317 Commit #19317											x

Task 19319 Commit #19317				x							
Task 19317 Datei – Vidierung.xlsx							x				
Requirement 19261: Vidierung & Rechte EVO Task 19317								x			
Requirement 19261 Task 19319 „alte BSZ übernehmen“ muss Rechte prüfen								x			
Task 19319 Datei – EvoFilter.vue				x							
Task 19319 Datei – Vidierung.xlsx									x		
Task 19319 Datei – EvoMigrationMode.vue											x
Task 19319 Datei – EvoEditMode.vue										x	
Task 19319 Datei – EvoEntiresWrapper.vue								x			
Datei – MigrationEntry.vue Datei – AddOrderLinesAside.vue				x							
Datei – OrderLineHeader.vue Datei – MigrationEntry.vue										x	
Datei – EvoMigrationMode.vue Datei – MigrationEntry.vue											x
TBWA EVO Übernahme aus Altfällen Task 19319											x
TBWA EVO Übername aus Altfällen Datei – EvoEditMode.vue										x	
TBWA EVO Übernahme aus Altfällen Datei - EvoMigrationMode.vue											x
Datei – EvoMigrationMode.vue Datei – TreatmentSetExtension.ts							x				

A.2 Eye Tracking Aufnahmen

Die verwendeten Aufnahmen sind auf dem beiliegenden USB-Stick zu finden. Da in den Aufnahmen sensible Patientendaten zu sehen sind, die nicht weitergegeben werden dürfen, sind nur die exportierten TSV-Dateien angehängt.

Literaturverzeichnis

[1] G. Spanoudakis, A. S. d. Garcez und A. Zisman, „Revising Rules to Capture Requirements Traceability Relations: A Machine Learning Approach," in *Proceedings of the Fifteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2003),* San Francisco Bay, CA, USA, 2003.

[2] M. Borg und P. Runeson, „IR in Software Traceability: From a Bird's Eye View," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, Baltimore, MD, USA, 2013.

[3] T. Li, S. Wang, D. Lillis und Z. Yang, *Combining Machine Learning and Logical Reasoning,* Beijing, China, 2020.

[4] J. Huffman Hayes, A. Dekhtyar und J. Osborne, „Improving Requirements Tracing via Information Retrieval," in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, Monterey Bay, CA, USA, 2003.

[5] F. Ungureanu, R. G. Lupu, A. Cadar und A. Prodan, „Neuromarketing and Visual Attention Study Using Eye Tracking Techiques," in *International Conference on System Theory, Control, and Computing (ICSTCC)*, Sinaia, Romania, 2017.

[6] M. A. Hassan, C. M. Aldridge, Y. Zhuang, X. Yin, T. McMurry, G. K. Rohde und A. M. Southerland, „Approach to Quantify Eye Movements to Augment Stroke Diagnosis With a Non-Calibrated Eye-Tracker," *IEEE Transactions on Biomedical Engineering,* Bd. 70, Nr. 6, pp. 1750-1757, 2023.

[7] S. Marx, G. Respondek, M. Stamlou, S. Dowiasch, J. Stoll, F. Bremmer, W. Oertel, G. Höglinger und W. Einhäuser, „Validation of mobile eye-tracking as novel and efficient means for differentiating progressive supranuclear palsy from Parkinson's disease," *Frontiers in Behavioral Neuroscience,* Nr. 6, 2012.

[8] A. Lev, Y. Braw, T. Elbaum, M. Wagner und Y. Rassovsky, „Eye Tracking During a Continuous Performance Test: Utility for Assessing ADHD Patients," *Journal of Attention Disorders,* Bd. 26, Nr. 2, pp. 245-255, 2022.

[9] Z. Sharafi, Z. Soh und Y.-G. Guéhéneuc, „A systematic literature review on the usage of eye-tracking in software," *Information and Software Technology,* Nr. 67, pp. 79-107, 2015.

[10] B. Sharif und T. Shaffer, „The Use of Eye Tracking in Software Development," in *Foundations of Augmented Cognition,* Los Angeles, CA, USA, 2015.

[11] B. Ramesh und M. Jarke, „Toward Reference Models for Requirements Traceability," *IEEE Transactions on Software Engineering,* Bd. I, Nr. 27, pp. 58-93, 2001.

[12] P. Mäder und A. Egyed, „Assessing the effect of requirements traceability for software maintenance," in *2012 28th IEEE International Conference on Software Maintenance (ICSM),* Trento, Italy, 2012.

[13] O. C. Z. Gotel und A. C. W. Finkelstein, „An Analysis of the Requirements Traceability Problem," in *Proceedings of IEEE International Conference on Requirements Engineering,* Colorado Springs, CO, USA, 1994.

[14] P. Arkley und S. Riddle, „Overcoming the Traceability Benefit Problem," in *2005 13th IEEE International Conference on Requirements Engineering (RE'05),* Paris, Frankreich, 2005.

[15] N. Ali, Z. Sharafi und Y.-G. Guéhéneuc, „An Empirical Study on Requirements Traceability Using Eye-Tracking," in *2012 28th IEEE International Conference on Software Maintenance (ICSM),* Trento, Italien, 2012.

[16] T. Shaffer, J. L. Wise, B. M. Walters, S. C. Müller, M. Flacone und B. Sharif, „iTrace: Enabling Eye Tracking on Software Artifacts within the IDE to Support Software Engineering Tasks," in *ESEC/FSE 2015: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering,* Bergamo, Italien, 2015.

[17] B. Walters, M. Falcone, A. Shibble und B. Sharif, „Towards an Eye-Tracking Enabled IDE for Software Traceability Tasks," in *2013 7th*

International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), San Francisco, CA, USA, 2013.

[18] B. Walters, T. Shaffer, B. Sharif und H. Kagdi, „Capturing software traceability links from developers' eye gazes," in *ICPC 2014: Proceedings of the 22nd International Conference on Program Comprehension*, Hyderabad, India, 2014.

[19] B. Sharif, J. Meinken, T. Shaffer und H. Kagdi, „Eye movements in software traceability link recovery," *Empirical Software Engineering*, Bd. III, Nr. 22, p. 1063–1102, 2017.

[20] B. Sharif und H. Kagdi, „On the Use of Eye Tracking in Software Traceability," in *TEFSE '11: Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, Waikiki, Honolulu HI USA, 2011.

[21] C. Rupp, „MASTER-Schablonen für Definition," in *Requirements-Engineering und -Management: Das Handbuch für Anforderungen in jeder Situation*, Carl Hanser Verlag, 2023, pp. 370-385.

[22] Statista Research Department, „statista," Statista GmbH, 29 02 2023. [Online]. Available: https://de.statista.com/statistik/daten/studie/828610/umfrage/marktanteile-der-fuehrenden-betriebssystemversionen-weltweit/. [Zugriff am 08 05 2023].

[23] IBM, „ibm.com," [Online]. Available: https://www.ibm.com/de-de/topics/docker. [Zugriff am 5 August 2023].

[24] N. Developers, „NumPy.org," NumPy, [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Zugriff am 03 08 2023].

[25] M. Ahrens, „Towards Automatic Capturing of Traceability Links by Combining Eye Tracking and Interaction Data," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, Zürich, Schweiz, 2020.