Wie können in Code Cities visualisierte Abläufe beim Programmverstehen unterstützen?

Bachelorarbeit

Alexander Kaiser

20. November 2023

Fachbereich 3 — Mathematik und Informatik
Studiengang Informatik

1. Gutachter: Prof. Dr. Rainer Koschke
2. Gutachter: Prof. Dr. Ute Bormann

ZUSAMMENFASSUNG

Diese Bachelorarbeit untersucht den Einfluss moderner Visualisierungstechniken in virtuellen Umgebungen auf das Verständnis von Software bei Softwareentwicklern. Im Zentrum steht die Analyse, wie eine weiterentwickelte Darstellung von Programmabläufen in der Softwarequalitätsanalysesoftware SEE im Vergleich zu traditionellen Methoden wie UML-Diagrammen auf Papier das Verständnis komplexer Softwareprozesse beeinflusst. Mittels eines sorgfältig konzipierten Experiments, in dem Teilnehmer zwei Projekte mit unterschiedlichen Visualisierungsmethoden bearbeiteten, wurden quantitative und qualitative Daten gesammelt. Diese Daten wurden anschließend mithilfe verschiedener Analysemethoden, wie der qualitativer Inhaltsanalyse, dem Chi-Quadrat-Tests und der deskriptiver Statistik ausgewertet.

Die Ergebnisse zeigen eine gemischte Reaktion auf die Verwendung von SEE. Während einige Teilnehmer von einem verbesserten Überblick und einer erhöhten Interaktivität profitierten, empfanden andere die virtuelle Umgebung als weniger intuitiv für das Verständnis der Programmabläufe. Interessanterweise war die Berufserfahrung der Teilnehmer kein signifikanter Faktor für die Unterschiede im Verständnis. Zusätzlich zu den Hauptergebnissen liefert die Arbeit Einblicke in die subjektiven Wahrnehmungen der Teilnehmer und gibt Vorschläge für Verbesserungen in SEE, was wertvolle Hinweise für zukünftige Entwicklungen in diesem Bereich bietet.

Diese Arbeit trägt damit zu einem tieferen Verständnis der Auswirkungen von Visualisierungstechniken in der Softwareentwicklung bei und zeigt sowohl die Potenziale als auch die Herausforderungen auf, die mit der Implementierung von virtuellen Visualisierungsmethoden verbunden sind. Die Ergebnisse und Erkenntnisse dieser Arbeit sind relevant für Softwareentwickler, Lehrkräfte im Bereich der Informatik und Entwickler von Lehrwerkzeugen, da sie interessante Ansätze für die Gestaltung effektiverer Lernumgebungen in der Softwareentwicklung bieten.

ERKLÄRUNG

Ich versichere, diese Arbeit — sofern dies nicht explizit anders gekennzeichnet wurde — ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 20. November 2023

Alexander Kaiser

DANKSAGUNG

Ich möchte mich bei Rainer Koschke bedanken, der auf jede meine zahlreichen Fragen immer eine passende Antwort parat hatte. Ich möchte mich außerdem bei Lisa Marie Voigt bedanken, die bereitwillig alle Versionen dieser Arbeit korrekturgelesen hat. Ebenfalls bedanke ich mich bei meinem Projektteam bei encoway, welches mir während der Verschriftlichung dieser Arbeit den Rücken freigehalten hat. Vielen Dank an die Probanden und alle Korrekturleser und Korrekturleserinnen.

GENDER-HINWEIS

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

INHALTSVERZEICHNIS

1 EINLEITUNG

Die moderne Softwareentwicklung steht vor zahlreichen Herausforderungen. Diese ergeben sich unter anderem aus der steigenden Komplexität von Softwareprojekten[22] und den hohen Anforderungen an Softwarequalität. Die Fähigkeit, den internen Ablauf von Softwareanwendungen klar zu verstehen und effektiv zu analysieren, ist von entscheidender Bedeutung. Fehler können schneller identifiziert, die Anwendungsperformance verbessert und die Wartbarkeit der Software erhöht werden [9]. Schon gute Dokumentation in Form von Texten und Diagrammen stellt für Entwickler eine enorme Zeitersparnis dar, weil sie sich dadurch einen guten Überblick über die Software verschaffen können[8]. Die Visualisierung einzelner Anwendungsbeispiele eines Programms erfolgt standardmäßig in der *Unified Modeling Language* (UML), oftmals in Form von Sequenzdiagrammen[13].

UML: *Die UML (Unified Modeling Language) definiert eine allgemein verwendbare Modellierungssprache (auch Notation genannt) [13]*

Diese Bachelorarbeit untersucht, wie sich eine weiterentwickelte Visualisierung von Programmabläufen innerhalb des Softwarequalitätsanalysetools „*Software Engineering Experience* (SEE)“ in *Code-City* auf das Programmverständnis von Softwareentwicklern gegenüber den traditionellen Methoden auswirkt.

SEE: *Eine interaktive Visualisierung von Software, welche die* Code-City- *Metapher verwendet und einen kollaborativen Multiplayer über verschiedene Plattformen.*

Code-City: *In der Code-City-Metapher werden Softwarekomponenten durch Gebäude in einer Stadt repräsentiert, wobei die Eigenschaften dieser Gebäude verschiedene Metriken der Software ausdrücken können — z. B. könnte die Höhe eines Gebäudes der Anzahl der Codezeilen entsprechen.*

1.1 HINTERGRUND DER STUDIE

Die Forderung nach besserer Dokumentation ist in der Softwareentwicklung weit verbreitet. [8] Nicht selten wurden aufgrund fehlender Dokumentation Projekte unbrauchbar[1]. Wissenstransfer ist für den Bestand eines Unternehmens unerlässlich. Entwickler müssen die Anwendung auf verschiedenen Ebenen verstehen, damit der geschriebene Code den Anforderungen entspricht. Unter anderem muss ein Programmierer verstehen, was der Anwender von der Applikation erwartet, welche Anwendungsfälle existieren und welche Klassen und Methoden schon vorhanden sind oder ähnliche Aufgaben übernehmen. Dafür bedarf es Softwareverständnis, das oftmals in Dokumentation oder Onboarding Programmen vermittelt wird. Die traditionellen Ansätze zur Verbesserung der Softwareverständlichkeit, wie Diagramme, haben ihre Grenzen, da sie nur den zweidimensionalen Raum nutzen und

1 https://www.tricentis.com/blog/real-life-examples-of-software-development-failures (letzter Aufruf 18.11.2023)

größtenteils statisch sind. In dieser Arbeit wird untersucht, ob diese Grenzen durch innovative digitale Werkzeuge innerhalb von virtuellen, dreidimensionalen Umgebungen so ergänzt werden können, dass sie das Programmverständnis verbessern.

1.2 FORSCHUNGSFRAGE UND AUFBAU DER ARBEIT

Die Frage, die innerhalb dieser Arbeit erforscht werden soll, ist „Wie können in Code Cities visualisierte Abläufe beim Programmverstehen unterstützen?“. Um diese Frage zu beantworten, werden zunächst in Kapitel 2 die in dieser Arbeit verwendeten Methoden beleuchtet. Danach wird in Kapitel 3 die Erweiterung der bestehenden Visualisierung thematisiert. Die Durchführung des Experiments wird in Kapitel 4 beschrieben. Die Auswertung der gesammelten Daten ist Thema des Kapitel 5, worauf sowohl das Fazit als auch der Zukunftsausblick des Kapitels 6 basieren.

2

METHODISCHE VORGEHENSWEISE

In diesem Kapitel wird das methodische Vorgehen zur Beantwortung der Leitfrage im Detail beschrieben. Als erstes wird in Kapitel 2.1 die Forschungsmethode erläutert, die in dieser Arbeit verwendet wird. Basierend darauf wird in Kapitel 2.2 die Konzeption des Fragebogens beschrieben. Kapitel 2.3 greift den Fragebogen auf und beschreibt, nach welchen Methoden die Antworten der Probanden ausgewertet werden. Am Ende dieses Kapitels gibt es eine Zusammenfassung in Form eines Zwischenfazits (2.4).

2.1 WAHL DER FORSCHUNGSMETHODE

Das Ziel dieser Studie besteht darin zu untersuchen, wie die entwickelte Visualisierung der Programmabläufe in Code Cities innerhalb von virtuellen Umgebungen, das Programmverständnis von Entwicklern beeinflusst. Um das zu untersuchen, muss das Programmverständnis durch SEE getestet werden. Die Ergebnisse wären ohne einen Vergleich mit dem Programmverständnis durch traditionelle Methoden, unbrauchbar. Es könnten schlichtweg keine Aussagen über die Beeinflussung von SEE auf das Softwareverständnis getroffen werden, da keine Basis gegeben ist. Daher werden zwei Projekte auf unterschiedliche Arten studiert: Projekt Eins wird mithilfe von Codeausschnitten und einem Sequenzdiagramm auf Papier präsentiert, während Projekt Zwei durch die entwickelte Darstellung in SEE visualisiert wird. Anschließend können basierend auf den Ergebnissen des Experiments fundierte Aussagen darüber getroffen werden,

1. wie die entwickelte Visualisierung in SEE im Gegensatz zu den traditionellen Methoden das Programmverständnis beeinflusst und

2. wie die Visualisierung subjektiv auf die Probanden wirkt.

Durch letzteres werden daraufhin Überlegungen getroffen, wie die Visualisierung zukünftig verbessert werden kann.

Die Wahrnehmung der Probanden während des Experiments innerhalb des virtuellen Raums sind vielschichtig, hoch komplex und

subjektiv, genauso wie in einer realen Umgebung[17]. Qualitative Forschungsmethoden bieten die nötige Flexibilität, um solch komplexe Phänomene zu erforschen. Während Quantitative Forschungsmethoden schneller die Frage nach der Auswirkung von SEE auf das Programmverständnis beantworten könnten, würden die subjektiven Eindrücke der Probanden nicht beachtet werden. Daher wird die qualitative Forschungsmethode in dieser Untersuchung bewusst gewählt, um die Eindrücke der Probanden zu analysieren. Genauer gesagt erlaubt die qualitative Forschung die Untersuchung der verschiedenen Einzelkomponenten auf das Programmverständnis der Probanden.

2.2 ENTWICKLUNG DES FRAGEBOGENS

2.2.1 *Einleitung*

Der Fragebogen wurde mit Microsoft Forms[1] erstellt und ist ein zentrales Instrument dieser Studie. Der Fragebogen zielt darauf ab, die Wahrnehmungen und Erfahrungen der Softwareentwickler in der virtuellen Lernumgebung zu erfassen. Die Gestaltung des Fragebogens basiert auf bewährten Forschungsprinzipien, die innerhalb der Unterkapitel erläutert werden und zielt darauf ab, umfassende und präzise Daten zu sammeln.

Der Fragebogen[2] besteht aus mehreren Teilen, die sich mit unterschiedlichen Aspekten der Erfahrung der Probanden befassen:

2.2.2 *Berufserfahrung*

Die Frage „Wie lange arbeitest Du schon als Softwareentwickler*in“ hilft dabei den Kontext der Probanden zu verstehen. Dies ist wichtig, um die Antworten im Licht der jeweiligen Berufserfahrung zu interpretieren, was eine gängige Praxis in der sozialwissenschaftlichen Forschung ist[2].

2.2.3 *Projektbezogene Fragen*

Die spezifischen Fragen zu den Programmabläufen zielen darauf ab, das Verständnis der Probanden für die einzelnen Abläufe zu beurteilen. Diese Art von Fragen spiegelt die Methodik der kognitiven Belastungstheorie wider. Diese geht davon aus, dass das Verständnis

1 https://www.microsoft.com/de-de/microsoft-365/online-surveys-polls-quizzes (zuletzt besucht 18.11.2023)

2 Der genaue Link zum Fragebogen: https://forms.office.com/pages/responsepage.aspx?id=6AzPNwmsfU6f8vH (zuletzt besucht am 20.11.2023)

komplexer Informationen besser erfasst werden kann, wenn spezifische, zielgerichtete Fragen gestellt werden[25].

2.2.4 *Bewertung der virtuellen Lernumgebung*

Die Fragen bezogen auf die Erfahrung innerhalb der virtuellen Umgebung sind darauf ausgelegt, die subjektive Wahrnehmung der Lernumgebung zu erfassen. Hierbei wurde sich für eine vierstufige Likert-Skala entschieden, da diese eine ausgewogene Balance zwischen einer zu simplen und einer zu komplexen Bewertungsskala bietet. Die vierstufige Likert-Skala zwingt die Probanden, ihre Einstellungen und Wahrnehmungen ohne eine neutrale Option auszudrücken. Dadurch müssen die Probanden eine tendenzielle Position einzunehmen. Das führt zu aussagekräftigeren und entscheidenderen Daten[5].

Allerdings hat die gewählte Entscheidung gegen eine neutrale Option auch ihre Nachteile. Sie könnte dazu führen, dass Probanden gezwungen sind, eine Meinung zu äußern, die sie eigentlich nicht haben, oder dass sie sich unwohl fühlen, wenn sie keine Option haben, die ihrer wahren Haltung entspricht. Daher ist es wichtig die Antworten im Kontext der restlichen Daten zu betrachten[5].

Diese Fragen folgen den Prinzipien der User-Experience-Forschung, die die Bedeutung des Benutzerfeedbacks zur Bewertung und Verbesserung technologischer Tools betont[1].

2.2.5 *Verbesserungsvorschläge*

Die Frage offene Frage „Wenn ich eine Funktionalität hinzufügen könnte, würde ich...“ gibt den Probanden die Möglichkeit, eigene Ideen und Vorschläge zu äußern. Diese offene Fragestellung ermöglicht es, unvorhergesehene Aspekte zu erfassen, die in standardisierten Fragen möglicherweise nicht berücksichtigt werden, weil die Fragen zu spezifisch sind[20]. Außerdem dient es erneut dem Zweck der Verbesserung von SEE, was teil der User-Experience-Forschung ist[1].

2.2.6 *Schlussfolgerung*

Der Fragebogen wurde so konzipiert, dass er sowohl qualitative als auch quantitative Daten liefert, die für die Beantwortung der Forschungsfragen dieser Studie wesentlich sind. Durch die Kombination von spezifischen, geschlossenen Fragen und offenen Antwortmöglichkeiten wird eine umfassende und vielschichtige Datengrundlage geschaffen. Außerdem vereinfacht die Verwendung von Microsoft Forms die Datenerhebung und -auswertung, indem die gesammelte Daten in unterschiedlichen Formaten präsentiert werden und einfach herunter-

ladbar sind.

2.3 ANALYSEMETHODEN

Dieses Kapitel bezieht sich auf die vier Themenbereiche innerhalb des Fragebogens. Es werden verschiedene Analysemethoden gegeneinander abgewogen, die für die Auswertung des Experiments verwendet werden.

2.3.1 *Projektbezogene Fragen*

Die in diesem Kontext geeignete Analysemethode ist die qualitative Inhaltsanalyse nach Mayring [18]. In der qualitativen Forschung bieten verschiedene Wissenschaftler unterschiedliche Ansätze der Inhaltsanalyse, die je nach Datenart variieren. Die Antworten auf die Fragen zu Projekt Eins und Projekt Zwei sind sehr spezifisch und erfordern einen strukturierten Ansatz, wie ihn die qualitative Inhaltsanalyse nach Mayring bietet. Krippendorffs Methode [16], obwohl sie flexibel und kontextuell stark ist, ist für diese Art von strukturierten Daten weniger geeignet. Ebenso sind Ansätze wie die Grounded Theory von Strauss und Corbin [23], die auf die Theoriebildung aus den Daten abzielen, oder die kritische Diskursanalyse von Fairclough [6], die sich auf die Beziehung zwischen Sprache und Macht fokussiert, für die spezifische Analyse zu explorativ. Mayrings Methode ermöglicht eine gezielte, inhaltsbezogene Auswertung, die optimal zu den klar definierten Antworten dieser Studie passt.

2.3.2 *Berufserfahrung*

In der Analyse des Einflusses der Berufserfahrung auf die Antwortqualität zu den Projektfragen wurde der Chi-Quadrat-Test als geeignetes Instrument ausgewählt. Alternative Methoden wie der T-Test oder die ANOVA erfordern normalverteilte, metrische Daten und wären daher für die durch die qualitative Inhaltsanalyse (Kapitel 2.3.1) kategorisierten Daten nicht passend [7]. Nichtparametrische Tests, wie der Mann-Whitney-U-Test, welcher für zwei unabhängige Stichproben geeignet ist, oder der Kruskal-Wallis-Test, der für mehr als zwei Gruppen geeignet ist, bieten zwar Flexibilität bei nicht normalverteilten Daten, sind jedoch für die Analyse von Häufigkeiten in kategorisierten Daten weniger geeignet[12] als der Chi-Quadrat-Test [19]. Der Chi-Quadrat-Test ist speziell für die Untersuchung von Häufigkeitsverteilungen in kategorisierten Daten konzipiert. Er setzt keine Normalverteilung voraus und ist leicht interpretierbar, was ihn für diese Arbeit besonders

geeignet macht. Somit bietet der Chi-Quadrat-Test die beste Kombination aus Anwendbarkeit auf die Datenstruktur und Einfachheit in der Durchführung und Interpretation für die Analyse des Zusammenhangs zwischen Berufserfahrung und Antwortqualität.

2.3.3 *Bewertung der Virtuellen Lernumgebung*

Für die Auswertung wurde entschieden, sich auf die deskriptive Statistik zu konzentrieren. Dieser Ansatz wurde gewählt, da er eine klare und direkte Übersicht über die zentralen Tendenzen und die Streuung der Daten bietet [5]. Obwohl Clusteranalysen tiefgehende Einblicke in die Daten ermöglichen, sind sie für die spezifische Fragestellung dieser Arbeit nicht unbedingt erforderlich. Clusteranalysen erfordern eine komplexe Datenstrukturierung und -interpretation [7]. Regressionsanalysen liefern ebenso wie Clusteranalysen genauere Einblicke, setzen allerdings statistische Annahmen voraus, die im Kontext einer Likert-Skala sehr problematisch sind [10].

2.3.4 *Vorschläge zur Verbesserung*

In Kapitel 2.3.1 wird argumentiert, dass die Antworten der Probanden zu spezifisch für eine Themenanalyse sind. In der letzten Frage nach zusätzlichen Funktionalitäten, sind die Antworten jedoch sehr offen und wenig spezifisch. Aus diesem Grund wird sich gegen eine qualitative Inhaltsanalyse und stattdessen für eine Themenanalyse entschieden. Die Diskussion für diese Entscheidung ist die gleiche, wie in Kapitel 2.3.1

2.4 ZWISCHENFAZIT

In diesem Kapitel wird ein umfassender Analyseansatz der Studiendaten zur virtuellen Lernumgebung dargestellt. Eine sorgfältige Auswahl von Forschungsmethoden wird getroffen, um die verschiedenen Aspekte der Studie effektiv zu beleuchten. Für die Analyse der projektbezogenen Fragen und der Auswirkungen der Berufserfahrung wird ein Mix aus qualitativer Inhaltsanalyse und dem Chi-Quadrat-Test gewählt, was eine tiefgehende und präzise Auswertung ermöglicht. Die Bewertung der virtuellen Lernumgebung selbst erfolgt durch deskriptive Statistik, um eine klare Übersicht über die allgemeinen Meinungstrends zu erhalten. Schließlich wird für die Analyse der Verbesserungsvorschläge eine Themenanalyse durchgeführt, um die offenen und vielfältigen Antworten der Probanden zu erfassen. Diese methodische Herangehensweise stellt sicher, dass die Studie ein umfassendes Bild von den Wahrnehmungen und Eindrücken der Probanden liefert und fundier-

te Erkenntnisse über die Effektivität der virtuellen Lernumgebung ermöglicht.

3 IMPLEMENTIERUNG DER ERWEITERUNG

Nachdem im vorherigen Kapitel die methodische Vorgehensweise dieser Arbeit erläutert wurde, widmet sich dieses Kapitel der Umsetzung der Erweiterung der Visualisierung von Programmabläufen.

3.1 PLANUNG

Innerhalb der Planung viel auf, dass vieles nicht neu entwickelt werden musste, sondern erweitert werden kann. Die Klassen `SEEJlgCity.cs` und `SEEJLGCityAnimation.cs` wurden innerhalb der Arbeit von Lennart Kipka[14] entwickelt und bieten einige nützliche Methoden zur Visualisierung von Programmabläufen innerhalb von SEE.

Allerdings galt es durch die Evolution des Projektes entstandene Fehler zu beheben. Der Code-Editor sowie die Übergangskanten wurden nicht mehr angezeigt, wodurch das gesamte Feature nicht nutzbar war.

Ebenfalls ist die Entwicklung einer audiovisuellen Informationsquelle Teil dieser Arbeit. Entwickler sollen Informationen zu Programmabläufen an bestimmten Übergängen einpflegen und abspielen können. Dies schafft nicht nur eine zusätzliche Möglichkeit Software zu dokumentieren, sondern auch eine weitere Art die Software zu erleben - per Gehör.

3.2 AKTUELLER STAND

In der Planungsphase fiel auf, dass bereits vieles durch die Arbeit von Lennart Kipka bestand (vgl. [14]). Leider gab es zum Zeitpunkt der Planung durch die stetige Evolution der Software seit Veröffentlichung der Bachelorthesis einige Programmfehler. Die von Lennart Kipka entwickelte Visualisierung der Komponentenaufrufe funktionierte aufgrund dessen nicht mehr. Innerhalb des Quellcodes war die Software jedoch ohne Probleme nachvollziehbar und ausführlich dokumentiert.

Aufbauend auf dieser Arbeit wurden Änderungen und Erweiterungen vorgenommen. Die Klassen `JLGCity.cs` und `JLGCityAnimation.cs`

wurden kopiert, um sicherzustellen, dass beide Entwicklungen in Zukunft innerhalb des Projektes existieren und zugänglich sind.

3.3 BUG FIXING

Zunächst wurde der bestehende Code in die Klassen `SEEJlgAudioCity.cs` `SEEJLGAudioCityAnimation.cs` kopiert und wieder funktional gemacht. Um möglichst effektiv vorzugehen, wurden Funktionen, wie das automatische Abspielen der Abläufe ausgebaut. Unter anderem diese Funktion führte zu den Fehlern, die der Entwicklungsstand mit sich brachte.

3.4 AUDIOVISUELLE ERWEITERUNG

Das Aufzeichnen von Sprache wurde innerhalb der Klasse `SEEJLGAudioCityAnimation.cs` hinzugefügt. Mittels einer Taste (standardmäßig F9) kann die Aufnahme zum aktuellen Ablauf gestartet bzw. gestoppt werden. Sobald ein Nutzer mit dem Pointer über die Code-City fährt bis zu einem Ablauf spult, der mit einer Audio-Datei hinterlegt ist, wird dieses Audio abgespielt. Sollte ein bereits vertonter Ablauf erneut vertont werden, wird die alte Datei überschrieben.

Float: *Ein Datentyp zur Darstellung von Gleitkommazahlen*

Die Dateien enthalten *Floats*, um das Audio zu speichern. Die Zuordnung zu den Abläufen geschieht mithilfe des `parsedJLG`(s.[14]) Parameters innerhalb der `SEEJLGAudioCityAnimation.cs`, der die Informationen zu allen Abläufen des Programms beinhaltet. Der Dateiname besteht aus dem Klassennamen, dem Methodennamen und dem aktuellen Index der *Java-Log* (LOC).

LOC: *LoC steht für „Line of Code“bzw. eine Zeile Quellcode*

Dadurch kann eine Zuordnung stattfinden. Bei jedem Vor- oder Zurückspulen eines Aufrufs, wird geprüft, ob eine Datei mit dem Namen, der sich aus den aktuellen Aufrufsdaten zusammensetzt, existiert. Sollte die Datei existieren, wird sie ausgelesen und mithilfe der `audioSource` abgespielt.

3.5 LIMITATIONEN

Innerhalb dieser Implementierung wurden nicht alle Funktionalitäten von Lennart Kipka repariert. Ebenfalls kommt es beim Abspielen der Audiodateien zu Fragmenten, was teilweise das Zuhören erschwert. Des weiteren würde eine Zuordnung nicht mehr funktionieren, nachdem sich die Aufrufe verändert haben. Ändert sich der Index der LOC, so muss eine neue Datei aufgenommen werden.

4

PLANUNG UND DURCHFÜHRUNG DES EXPERIMENTS

.

4.1 ENTWICKLUNG DER MATERIALIEN

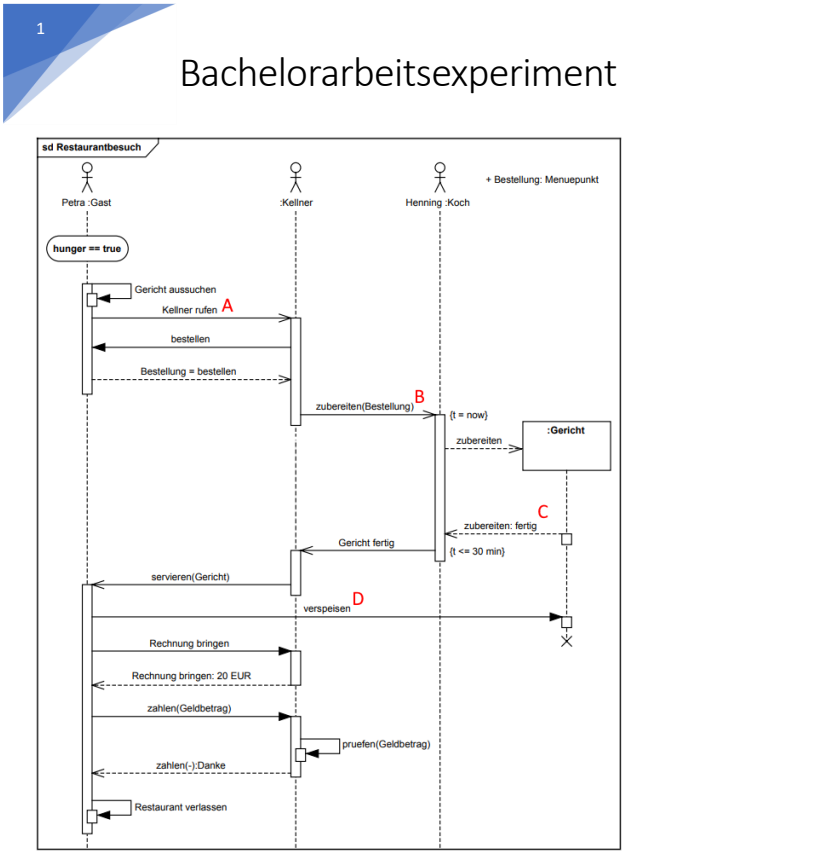
4.1.1 *Beispielprojekt 1*

Für die Untersuchung der Fragestellung ist es wichtig, dass die in Durchgang Eins und Zwei vorgestellten Projekte eine ähnliche Komplexität besitzen. Eine signifikant höhere Anzahl an Klassen oder Methodenaufrufen wäre nicht repräsentativ, da die Chance, dass die Probanden komplexer Projekte weniger verstehen, höher ist. Christoph Kechers Buch „UML 2 - Das umfassende Handbuch“ enthält ein sehr einfaches, alltägliches und gut verständlich Sequenzdiagramm[4]. Dieses ist auf der Abbildung 4.1 zu sehen. Das Sequenzdiagramm wird zur weiteren Vereinfachung synchronisiert. Das bedeutet, dass alle asynchronen Aufrufe, welche nicht direkt am kritischen Pfad beteiligt sind, entfernt werden. Das vereinfacht die Analyse in SEE, da Nebenläufigkeit der Aufrufe nur eine weitere Komplexitätsschicht innerhalb der virtuellen Lernumgebung darstellt, die in dieser Arbeit nicht betrachtet werden soll. Ebenfalls wurden an vier Stellen Kommentare zum Programmablauf eingefügt. Der Grund dafür lag in der Neuentwicklung innerhalb von SEE, bei der audiovisuelle Übergänge implementiert wurden, die zuvor aufgezeichnete Kommentare enthalten. Diese Übergänge spielen die Nachrichten beim Erreichen des Aufrufs ab, wodurch die Nutzer zusätzlich zu visuellen Informationen auch sprachbasierte Inhalte erhalten. Damit jedoch bei beiden Durchgängen die gleichen Informationen für die Probanden enthalten sind, wurden diese bei Projekt Eins in Form von Kommentaren an das Sequenzdiagramm geschrieben. So, wie das Sequenzdiagramm die Abläufe auf traditionelle Art verdeutlicht, so verdeutlichen die Kanten die Sequenzen, bzw. Aufrufe in SEE. An beiden werden extra Informationen gepflegt. Somit scheint auch der Ort, an denen die zusätzlichen Inhalte gepflegt wurden gleich. Ebenfalls wurde der Code im *Dark Theme* von *IntelliJ IDEA* zur Verfügung gestellt, da die meisten Probanden in ihrer eigenen (IDE) die gleiche Einstellung nutzen. So mussten sie sich nicht umstellen.

Dark Theme*: Der "Dark Mode" (oder Dunkelmodus) ist eine Benutzeroberflächeneinstellung oder ein Design, das dunkle Farben für Hintergründe und helle Farben für Texte und andere Elemente verwendet.*

IntelliJ IDEA*: IntelliJ ist eine von JetBrains entwickelte Entwicklungsumgebung für mehrere Programmiersprachen.*

IDE*: IDE steht für „Integrated Development Environment“, was auf Deutsch „Integrierte Entwicklungsumgebung“ bedeutet. Innerhalb von Entwicklungsumgebungen schreiben Entwickler Software, da sie viele nützliche Funktionen beinhalten.*



A – Der Kellner wird gerufen, indem der Methode der Kellner übergeben wird. Im nächsten Schritt wird dann bestellt.

B – Der Kellner ruft die Methode „zubereiten“ auf dem Objekt des Kochs auf. Dieser startet nun die Schleife

C – Der Koch hat das Gericht fertig zubereitet. Dies hat statt 30 Minuten im Programm nur 30 Sekunden gedauert. Jede Schleifeniteration lässt den Thread 1 Sekunde schlafen

D – Das Verspeisen wird simuliert, indem der Java Garbage Collector das Objekt einfach aufräumt

Abbildung 4.1: Angepasstes Sequenzdiagramm des Restaurant Szenarios aus „UML 2 - Das umfassende Handbuch“ S. 381

Der Code wird auf Papier ausgedruckt. Der Code und die eine Seite des Sequenzdiagramms umfassen zusammen 8 Seiten, was das gesamte Material zu Projekt Eins umfasst. Dieses Material ist im Anhang zu finden.

4.1.2 *Beispielprojekt 2*

Um eine ähnliche Komplexität, wie die des Beispielprojekt Eins zu gewährleisten, wird das Restaurant aus dem Beispiel zu einer Autowerk-

statt. In Abbildung 4.2 ist das angepasste Sequenzdiagramm zu sehen. Die Klassen-, Variablen- und Methodennamen, sowie Kommentare wurden an das Beispiel aus Abbildung 4.1 angepasst. Auch ein paar Aufrufe wurden umstrukturiert, damit der Zusammenhang nicht direkt besteht und im Versuch möglich kein Lernprozess zwischen beiden Projekten entsteht, da ansonsten die Ergebnisse verfälscht wären, weil in Durchgang Zwei nicht nur das Softwareverständnis zu Projekt Zwei geprüft werden würde, sondern das zu Projekt Eins.

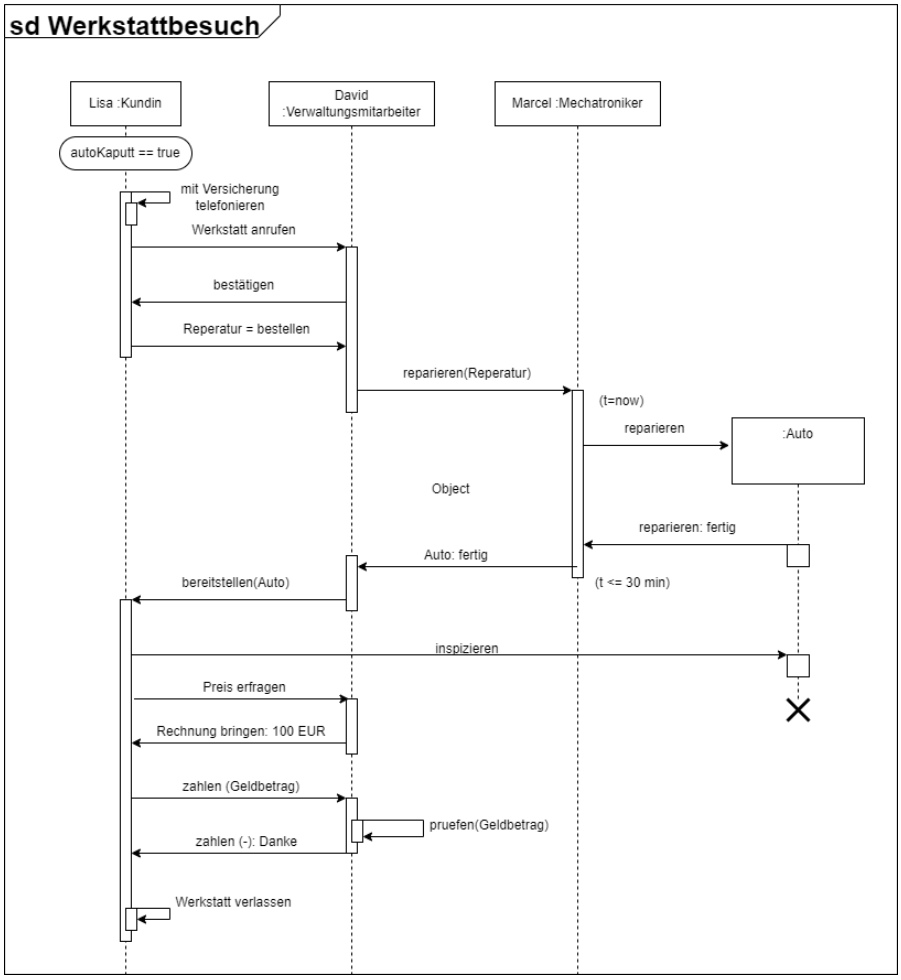


Abbildung 4.2: Sequenzdiagramm des abgeänderten Werkstatt Szenarios

Beide Projekte sind in Java geschrieben und sind im Anhang verfügbar.

4.1.3 *Einpflegen in SEE*

Damit das Beispielprojekt Zwei in SEE integriert werden kann, bedarf es einer *Graph Exchange Language* (GXL)-Datei. SEE nutzt diese Dateien, um Softwareprojekte in Form von Code Cities darzustellen (vgl. [15]).

GXL: *standardisiertes Format zum Austausch von Graphdaten. Es wird häufig in der Softwaretechnik verwendet, insbesondere im Bereich der Softwarevisualisierung und des Reverse Engineering*

Diese Datei wurde freundlicherweise von Prof. Dr. Rainer Koschke zur Verfügung gestellt, nachdem er das zweite Beispielprojekt erhielt. Diese Datei wurde mithilfe der *Axivion Suite* erstellt, welche die Informationen zum Projekt Zwei in Graphenform besitzt.

Axivion Suite: *Die Axivion Suite ist ein umfassendes Software-Tool, das mehrere Schlüsselfunktionen besitzt, um die Qualität und Zuverlässigkeit von Softwareprojekten sicherzustellen*

JLG: *JLG steht für Java-Log und beinhaltet Informationen zum Ablauf eines Java-Programms (vgl. [14])*

Makefile: *Eine Makefile ist eine spezielle Datei, die häufig in der Softwareentwicklung verwendet wird, um den Build-Prozess eines Projekts zu automatisieren*

Zusätzlich zu der Darstellung in Form von Code-Cities bedarf es allerdings noch Informationen über die Aufrufe. Diese Informationen werden in Form von *Java-Log* (JLG) gespeichert. Wie bereits vorher erwähnt, baut diese Arbeit auf der Arbeit von Lennart Kipka auf. Daher gab es bereits eine *Makefile* zum Erstellen der JLG ().

Nachdem sowohl GXL und JLG Dateien erstellt waren, konnte das Projekt in SEE angezeigt werden. Wie das aussieht, ist in 4.3 zu sehen.

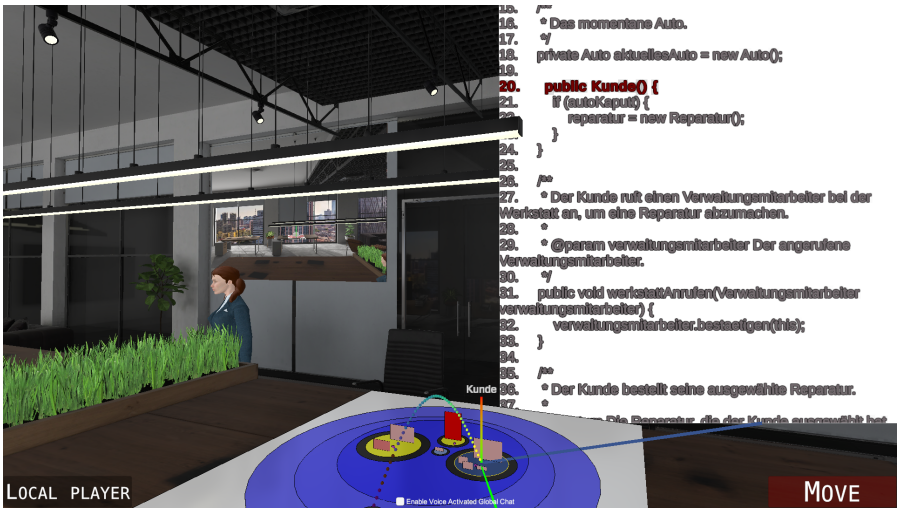


Abbildung 4.3: Visualisierung des Projekts und der Aufrufe in Code Cities

4.2 STICHPROBENAUSWAHL

Die Stichprobenauswahl des Experiments erfolgte gezielt unter Softwareentwicklern, die alle innerhalb eines renommierten Bremer Softwareunternehmens tätig sind. Die Entscheidung, ausschließlich auf Probanden aus diesem Unternehmen zurückzugreifen, basiert auf zweierlei Gründen:

- Die Softwareentwickler sind bereits ausgebildet und erfahren, wodurch sie alle nötigen Vorkenntnisse besitzen, die für das Experiment essentiell sind.
- Durch die Nähe zum Unternehmen kann auf mehr Probanden zurückgegriffen werden.

Im Rahmen der Überlegungen für die Bachelorarbeit wurde in Betracht gezogen, ob es von Vorteil wäre, wenn alle Probanden aus

demselben Unternehmen stammen und an ähnlicher Software aus einer bestimmten Domäne arbeiten. Es wurde jedoch festgestellt, dass dies für die Beantwortung der Forschungsfragen nicht von entscheidender Bedeutung ist. Die Probanden waren weder mit den beiden untersuchten Projekten noch mit SEE vertraut. Die einzige Gemeinsamkeit bestand darin, dass sie täglich in Java programmierten. Dies wurde jedoch als vorteilhaft für das Experiment angesehen, da in einem realen Anwendungsfall die Entwickler, die das Tool nutzen würden, die Programmiersprache in den meisten Fällen gut kennen sollten.

4.3 VERSUCHSABLAUF

Nach Betreten des Raumes werden die Probanden begrüßt und mit dem Versuchsablauf vertraut gemacht.

Das Experiment gliedert sich in zwei wesentliche Phasen:

In der ersten Phase erhalten die Probanden alle Materialien für das erste Beispielprojekt in Papierform. Sie haben fünf Minuten Zeit, um sich mit dem Material vertraut zu machen, bevor es ihnen entnommen und durch den ersten Fragebogen ersetzt wird. Für das Ausfüllen dieses Fragebogens stehen zehn Minuten zur Verfügung, nach deren Ablauf die Probanden gebeten werden, den ausgefüllten Fragebogen abzugeben.

Die zweite Phase beginnt mit der Bereitstellung eines Laptops, auf dem die gebaute SEE Software bereits installiert und einsatzbereit ist. Nach einer kurzen Einführung in die Software haben die Probanden erneut fünf Minuten Zeit, um sich das zweite Beispielprojekt, das in SEE dargestellt wird, anzusehen und zu verstehen. Nach dieser Zeitspanne wird der Laptop wieder entnommen und durch den zweiten Fragebogen ersetzt. Auch für diesen Teil des Experiments wird den Probandinnen und Probanden eine Bearbeitungszeit von zehn Minuten eingeräumt. Nach Ablauf dieser Zeit wird der zweite Fragebogen eingesammelt, wonach das Experiment beendet ist.

4.4 ETHIK UND DATENSCHUTZ

Die Datenerhebung für dieses Experiment erfordert keine personenbezogenen Daten der Probanden. Es werden keinerlei personenbezogene Informationen wie Namen, Adressen, Kontaktdaten oder persönliche Identifikationsmerkmale während des Experiments erfasst, gespeichert oder verwendet. Alle Informationen, die im Rahmen des Experiments

gesammelt werden, dienen ausschließlich der Auswertung der Forschungsergebnisse und werden vertraulich behandelt.

4.5 DURCHFÜHRUNG DES EXPERIMENTS

Die Durchführung des Experiments erfolgte in einer kontrollierten Umgebung, innerhalb eines kleinen Raumes im Büro des genannten renommierten Softwareunternehmens, wie in 4.4 zu sehen ist. Die Probanden wurden vor Beginn des Experiments über den Ablauf informiert. Sie hatten vor Beginn der zweiten Lernzeit Zeit, sich mit SEE vertraut zu machen. Ihnen wurden einige nützliche Short Cuts, wie beispielsweise das Aufrufen des Aktionsmenüs präsentiert.

Diese Einführung in SEE war entscheidend, um eine Beeinträchtigung der Ergebnisse durch Unvertrautheit mit der Software zu vermeiden und somit eine Verfälschung der Daten auszuschließen. Die Bedienung von SEE unterscheidet sich von der Steuer vieler Ego-Perspektiv-Spiele, da sie auch für Virtual Reality konzipiert wurde, was auf den ersten Blick zu Unklarheiten führen kann. Um sicherzustellen, dass die Antworten der Probanden ausschließlich auf die Neuartigkeit der Visualisierung und nicht auf Schwierigkeiten mit der Bedienung oder dem Interface zurückzuführen sind, wurde eine umfassende Einführung in SEE gegeben. Da alle Probanden bereits umfangreiche Erfahrungen mit der Arbeit auf Papier, einschließlich der Code-Analyse, hatten, war es wichtig, ihnen trotz der begrenzten Zeit in SEE eine solide Grundlage zu bieten, um vergleichbare Bedingungen zu schaffen.

Es wurde abgewogen, ob dieses Experiment zweistufig oder mit zwei verschiedenen Personengruppen durchgeführt werden sollte. Letztlich wurde sich jedoch für ein zweistufiges Experiment entschieden, da so die subjektiven Eindrücke der Probanden auf die neue Visualisierung gesammelt werden konnten. Diese Aussagen sind für die Auswertung und den Ausblick interessant, weil sie einerseits mit der Antwortqualität der Probanden korrelieren könnte und andererseits die Verbesserungsvorschläge in der Zukunft zu einer verbesserten Visualisierung von Aufrufen in Code Cities führen könnten.

Abbildung 4.4: Ein Proband nimmt am Experiment teil und liest die Materialien des ersten Projekts durch

5 AUSWERTUNG

Kurze Einführung, wie die Auswertung von statten geht Mehrere Stufen innerhalb des Fragebogens: Projektverständnis, subjektive Eindrücke, Anmerkungen

5.1 EINLEITUNG

Im Rahmen dieser Auswertung wird eine systematische Analyse der gesammelten Daten vorgenommen, um die Leitfrage zu beantworten. Die Analyse ist in vier Kategorien gegliedert, die jeweils unterschiedliche Aspekte der Nutzung und Wahrnehmung von Code Cities und deren Einfluss auf das Programmverstehen.

Im ersten Kapitel, 5.2 Projektbezogene Fragen, werden die Verständnisse zu beiden Projekte erläutert. Dadurch werden die Kenntnisstände der zu beiden Projekten klar. Basierend darauf können Aussagen gemacht werden, wie die Visualisierung innerhalb von SEE das Softwareverständnis beeinflusst.

Im darauffolgenden Kapitel, 5.3 Berufserfahrung, wird untersucht, wie die berufliche Vorerfahrung der Probanden ihr Verständnis und ihre Interaktion mit Code Cities beeinflusst. Diese Analyse ist von Bedeutung, da sie Aufschluss darüber gibt, ob das Softwareverständnis in SEE mit der Berufserfahrung zusammenhängt.

Das vierte Kapitel, 5.4 Bewertung der Virtuellen Lernumgebung, zielt darauf ab, die allgemeine Effektivität und Akzeptanz von SEE als Werkzeug zur Visualisierung von Programmabläufen zu bewerten. Durch die Analyse der Reaktionen und Bewertungen der Probanden wird ein tieferes Verständnis darüber erlangt, wie die virtuelle Umgebung das Lernen und Verstehen beeinflusst und welche Aspekte besonders wertvoll oder verbesserungswürdig sind.
Im vorletzten Kapitel, 5.5 Vorschläge zur Verbesserung der Lernwerkzeuge, wird die Verbesserungswürdigkeit genauer analysiert. Die Ideen der Nutzer werden gesammelt, welche entscheidend für die zukünftigen Entwicklungen und Anpassungen von SEE sind.

Insgesamt bildet die Analyse der vier Kategorien im Fragebogen das Fundament, um umfassend zu erörtern, wie die Entwickelte Visualisie-

rung als neue innovative Methode das Verständnis von Softwareabläufen verbessern kann. Die Erkenntnisse aus jeder Kategorie tragen dazu bei, das Gesamtergebnis am Ende zu interpretieren.

5.2 PROJEKTBEZOGENE FRAGEN

5.2.1 *Einleitung*

Im folgenden Kapitel werden die Erkenntnisse aus den Antworten auf die projektbezogenen Fragen, die im Rahmen des Experiments gestellt wurden, detailliert analysiert und diskutiert. Diese Analyse gliedert sich in zwei Hauptteile, basierend auf den beiden unterschiedlichen Projekten, die von den Probanden studiert wurden. Jedes Projekt wurde unter verschiedenen Bedingungen und mit unterschiedlichen Hilfsmitteln verstanden, um ein tiefgreifendes Verständnis dafür zu entwickeln, wie die beiden Ansätze das Programmverstehen beeinflussen.

Durch den Vergleich der Ergebnisse ist es möglich, fundierte Rückschlüsse darüber zu ziehen, wie die Lernumgebungen und Werkzeuge das Programmverstehen beeinflussen. Ziel ist es, zu verstehen, ob die weiterentwickelte Visualisierung in SEE positiv zum Verständnis des Programmablaufs beiträgt.

Der Vergleich wird gemacht, nachdem sukzessiv die verschiedene Analyseschritte durchgeführt wurden. Zunächst werden die Antworten der Probanden in vorher definierte Kategorien eingeordnet und somit quantifiziert. Danach werden die zuvor eingeordneten Antworten der beiden Projekte verglichen und basierend darauf wird in Kapitel 5.2.3 das Ergebnis diskutiert.

5.2.2 *Ergebnisse*

Vor dem Hintergrund, dass es im durchgeführten Experiment um Informationsaufnahmen geht, werden im Folgenden, gemäß Mayrings Methodologie, drei Kategorien definiert, die unter anderem durch Literatur aus dem Bildungssektor zu begründen sind:

1. Korrektheit:
   Eine Antwort wird als „korrekt“ gewertet, wenn sie genau mit den in den Projekten implementierten Abläufen und Strukturen übereinstimmt. Dieses Kriterium spiegelt ein tiefes Verständnis und eine genaue Kenntnis der Software wider. Genauigkeit und ein vollumfängliches Verständnis eines Projektes sind essentiell für das Schreiben von qualitativem Code, wie von Steven McConnell [24] in seinem Buch „Code Complete“ beschrieben. Daher kann durch diese Kategorie das vollumfängliche Verständnis der

Probanden bewertet werden.

2. Teilweise Korrektheit:
   Eine Antwort gilt als „teilweise korrekt“, wenn sie wesentliche Aspekte der Fragestellung trifft, aber wichtige Details auslässt oder fehlerhaft wiedergibt. Diese Kategorie erkennt an, dass die Probanden wichtige Aspekte der Fragestellung verstehen, aber teilweise Details auslassen oder bestimmte Implementierungen nicht vollständig erfassen. Somit ist zwar kein vollständiges, aber ein grundlegend solides Wissen vorhanden, was im Werk „Teaching for Quality Learning at University“ von John Biggs und Catherine Tang [3] als erster Schritt zur Förderung des Lernens beschrieben wird. Durch diese Kategorie können Antworten, die die grundlegenden Abläufe und Strukturen verstehen, aber nicht ausreichen, um als korrekt zu gelten, von den inkorrekten Antworten abgegrenzt werden. Durch die detaillierte Betrachtung sind auch die Ergebnisse granulärer, wodurch die Interpretationen genauer sind.

3. Inkorrektheit:
   Eine Antwort wird als „inkorrekt“ bezeichnet, wenn sie keine oder eine falsche Darstellung der in den Projekten implementierten Abläufe und Strukturen bietet. Diese Kategorie ist wesentlich, um Missverständnisse oder Lücken im Verständnis der Probanden zu identifizieren. In der Didaktik ist die Einteilung und Adressierung von Fehlern ein sehr wichtiges Feedback, wie im Artikel „The Power of Feedback“ von John Hattie und Helen Timperley [11] beschrieben.

Insgesamt ermöglichen diese drei Kategorien eine umfassende und differenzierte Bewertung der Antworten. Sie umfassen sowohl die Komplexität, der Strukturen der Projekte im Experiment als auch die verschiedenen Verständnisebenen einer Bildungsumgebung.

Philipp Mayring führt in seinem Buch „Qualitative Inhaltsanalyse: Grundlagen und Techniken“ an:

> Die Inhaltsanalyse ist kein Standardinstrument, das immer gleich aussieht; sie muss an den konkreten Gegenstand, das Material angepasst sein und auf die spezifische Fragestellung hin konstruiert werden.[21]

Aus diesem Grund wird nachfolgend, obwohl nicht in Mayrings Methodologie beschrieben, für jede Frage eine klare Definition für die Kategorisierung der Antworten eingeführt. So ist die Einteilung der Antworten klar nachzuvollziehen und erscheint weniger subjektiv.

- Projekt 1: Restaurant-Szenario
  - **Welche Klasse(n) ist/sind dafür zuständig, das Gericht zuzubereiten?**
    - Korrekt: Koch
    - Teilweise korrekt: Antwort beschreibt die Klasse Koch.
  - **Was ist der nächste Aufruf, nachdem der Gast den Kellner gerufen hat?**
    - Korrekt: Der Kellner ruft „bestellen“ auf.
    - Teilweise korrekt: Antwort bezieht sich auf die Interaktion zwischen Gast und Kellner, ohne die spezifische Methode zu nennen.
  - **Was ist der nächste Aufruf, nachdem der Kellner dem Gast das Essen gebracht hat?**
    - Korrekt: Der Gast ruft „verspeisen“ auf.
    - Teilweise korrekt: Beschreibt einen Teil des Essensvorgangs.
  - **Beschreibe den Zweck der Schleife innerhalb der Klasse „Koch“.**
    - Korrekt: Die Schleife simuliert die Zubereitungszeit des Gerichts von 30 Sekunden oder 30 Iterationen.
    - Teilweise korrekt: Erwähnung der Zubereitungssimulation.
  - **Was passiert, nachdem der Gast das Restaurant verlassen hat?**
    - Korrekt: Das Programm gibt „Tschüss“ aus.
    - Teilweise korrekt: Das Programm endet.
  - **Wie viele Methoden hat die Kochklasse und was machen diese?**
    - Korrekt: Eine Methode, „zubereiten“, die das Gericht zubereitet.
    - Teilweise korrekt: Erwähnung der Methode „zubereiten“ oder „1“ ohne vollständige Beschreibung ihrer Funktion.
- Projekt 2: Werkstatt-Szenario
  - **Welche Klasse(n) ist/sind dafür zuständig, die Reparatur auszuführen?**
    - Korrekt: Mechatroniker
    - Teilweise korrekt: Antwort beschreibt die Klasse Mechatroniker.
  - **Was passiert, nachdem der Kunde die Bestätigung von der Verwaltungsmitarbeiterin erhält?**

    * Korrekt: Der Mechatroniker repariert das Auto.
    * Teilweise korrekt: Beschreibt teilweise den Reparaturprozess.
  - **Was ist der nächste Aufruf, nachdem die Verwaltungsmitarbeiterin dem Kunden das Auto bereitstellt?**
    * Korrekt: Der Kunde „inspiziert“ das Auto.
    * Teilweise korrekt: Umschreibt den Inspektionsschritt
  - **Beschreibe den Zweck der Schleife innerhalb der Klasse „Mechatroniker“.**
    * Korrekt: Die Schleife simuliert die Reparatur des Autos von 30 Sekunden oder 30 Iterationen.
    * Teilweise korrekt: Erwähnung der Reparatursimulation.
  - **Was geschieht, nachdem der Kunde die Werkstatt verlassen hat?**
    * Korrekt: Das Programm gibt „Tschüss“ aus.
    * Teilweise korrekt: Das Programm endet.
  - **Wie viele Methoden hat die Mechatroniker-Klasse und was bewirken diese?**
    * Korrekt: Eine Methode, „reparieren“, die das Auto repariert.
    * Teilweise korrekt: Erwähnung der Methode „reparieren“ oder „1“ ohne vollständige Beschreibung ihrer Funktion.

Alle Antworten, die nicht anhand der Definitionen in die Kategorien eingeordnet werden können, gelten als „inkorrekt“.

Im nächsten Schritt werden die Antworten mithilfe der Definitionen kodiert (vgl. [18]). Dies ist wichtig, um die Antworten in eine quantifizierbare Struktur zu bringen. Nur so können die Antworten im Hinblick auf die beiden Projekte verglichen werden.

Zusammengefasst ergeben sich dadurch in Projekt 1 34 korrekte, 23 teilweise korrekte und 9 inkorrekte Antworten. In Projekt 2 gibt es 35 korrekte, 6 teilweise korrekte und 25 inkorrekte Antworten.

5.2.3 *Interpretation*

Die Daten legen nahe, dass sowohl die traditionelle Methode (Projekt 1) als auch die innovative Visualisierung in SEE (Projekt 2) effektiv waren, um das Verständnis von Softwarestrukturen zu vermitteln. Allerdings zeigen die unterschiedlichen Muster in den Antwortkategorien, dass jede Methode ihre eigenen Herausforderungen und Vorteile mit sich bringt. Die höhere Anzahl an inkorrekten Antworten in Projekt 2 könnte ein Anlass sein, die Gestaltung und Benutzerfreundlichkeit von

SEE näher zu untersuchen und möglicherweise anzupassen, um die Verständlichkeit und Intuitivität zu verbessern.

5.3 BERUFSERFAHRUNG

5.3.1 *Einleitung*

In diesem Abschnitt wird der Einflusses der Berufserfahrung auf die Antwortqualität in zwei unterschiedlichen Softwareverständnis-Experimenten untersucht. Diese Analyse basiert auf der Hypothese, dass berufliche Erfahrungen das analytische Verständnis und die Fähigkeit, Softwarekonzepte zu interpretieren, beeinflussen können.

5.3.2 *Ergebnisse*

Die Ergebnisse werden in Form von Kreuztabellen dargestellt, die die Anzahl der Antworten in jeder Kategorie nach Berufserfahrung zeigen.

Projekt 1:

Berufserfahrung	Inkorrekt	Korrekt	Teilweise Korrekt
3 - 5 Jahre	4	18	8
< 3 Jahre	4	11	9
> 5 Jahre	1	5	6

Chi-Quadrat-Wert: 2.62 p-Wert: 0.623
Projekt 2:

Berufserfahrung	Inkorrekt	Korrekt	Teilweise Korrekt
3 - 5 Jahre	14	13	3
< 3 Jahre	8	16	Keine
> 5 Jahre	3	6	3

Chi-Quadrat-Wert: Kann nicht berechnet werden p-Wert: Kann nicht berechnet werden

Da es bei Projekt 2 keine Antwort gibt, von den Probanden, die weniger als drei Jahre als Softwareentwickler arbeiten, die teilweise korrekt ist, kann der Chi-Quadrat-Wert nicht ausgerechnet werden. Zumindest wird in Fachliteratur, wie beispielsweise in * Buch „“ darauf hingewiesen, dass das Ersetzen von fehlenden Werten durch eine 0 gerade bei so einer geringen Probandenanzahl, die Ergebnisse beeinflussen kann.

Unter Berücksichtigung dessen wird im folgenden der Eintrag „Keine“ durch „0“ in Projekt 2 ersetzt:

Berufserfahrung	Inkorrekt	Korrekt	Teilweise Korrekt
3 - 5 Jahre	14	13	3
< 3 Jahre	8	16	0
> 5 Jahre	3	6	3

Daraus ergibt sich ein Chi-Quadrat-Wert von 8.21 und ein p-Wert von 0.084.

5.3.3 *Interpretation*

Bei Projekt 1 weist der hohe p-Wert darauf hin, dass keine signifikanten Unterschiede in der Antwortqualität zwischen den Berufserfahrungsgruppen bestehen. Dies bedeutet, dass die Art der Aufgabenstellung in

Projekt 1 unabhängig von der Berufserfahrung der Probanden verstanden wurde.

Dieses Ergebnis ist intuitiv, da bei der Komplexität von Projekt 1 weder ein Junior, noch ein Intermediate oder Senior Entwickler Schwierigkeiten haben sollte. Besonders, da dieses Projekt mithilfe der gewohnten Materialien, wie sie zum Beispiel bei der Berufsausbildung zum Fachinformatiker Anwendungsentwicklung benutzt werden, studiert wurde.

Nach der Anpassung des fehlenden Wertes zu 0, liegt der p-Wert nahe am Signifikanzniveau, was auf einen möglichen, aber nicht statistisch signifikanten Trend hinweist. Es könnte bedeuten, dass in Projekt 2 die Berufserfahrung einen gewissen Einfluss auf die Antwortqualität hat. Allerdings reicht der Wert nicht aus, um konkrete Schlüsse aus dem Ergebnis zu ziehen. Mit Hinblick auf den literarischen Hinweis, dass eine Anpassung von fehlenden Werten die Ergebnisse beeinflussen könnte, wird das Ergebnis so interpretiert, wie beim Projekt 1.

Es kann mit angrenzender Sicherheit gesagt werden, dass die Berufserfahrung keinen Einfluss auf die Antwortqualität in dem durchgeführten Experiment hat.

5.4 BEWERTUNG DER VIRTUELLEN LERNUMGEBUNG

5.4.1 *Einleitung*

In diesem Kapitel liegt der Fokus auf der Bewertung der virtuellen Lernumgebung, speziell im Kontext von SEE, die im zweiten Projekt des Experiments verwendet wurde. Die Beurteilung der Effektivität und des Einflusses dieser neuen Lernumgebung auf das Programmverstehen steht im Mittelpunkt der Analyse. Ziel ist es, zu verstehen, wie die Probanden die virtuelle Lernumgebung subjektiv beantworten.

5.4.2 *Ergebnisse*

Die deskriptive Analyse ergibt folgendes:

- Frage: „Die virtuelle Umgebung ermöglicht mir einen besseren Überblick“
  - Anzahl der Antworten: 11
  - Häufigste Antwort: „Eher nicht “ (5 Mal gewählt)
  - Weitere Antworten: „Stimmt “ (4 Mal), „Nein “ (2 Mal)

- Frage: „Durch eine virtuelle Umgebung konnte ich das Programm leichter nachvollziehen“
  - Anzahl der Antworten: 11
  - Häufigste Antwort: „Eher nicht “ (5 Mal gewählt)
  - Weitere Antworten: „Stimmt “ (4 Mal), „Nein “ (2 Mal)

5.4.3 *Interpretation*

Die Ergebnisse zeigen, dass die Meinungen zur virtuellen Lernumgebung gemischt sind. Während bei der ersten Frage die Zustimmung („Stimmt “) überwiegt, wird bei der zweiten Frage die Option „Eher nicht “ häufiger gewählt. Dies deutet darauf hin, dass die Probanden zwar einen besseren Überblick in der virtuellen Umgebung sehen, aber das Programmverständnis nicht unbedingt als erleichtert empfinden. Die Nichtauswahl von „Stimmt absolut“ weist auf Raum für Verbesserungen hin oder darauf, dass die Vorteile von SEE nicht für alle Probanden überwiegen.

5.5 VORSCHLÄGE ZUR VERBESSERUNG DER LERNWERKZEUGE

5.5.1 *Einleitung*

In diesem Kapitel wird die Analyse der aus dem Experiment gesammelten Vorschläge zur Verbesserung der Lernwerkzeuge vorgenommen. Diese Vorschläge sind im Anhang verfügbar.

Die in diesem Kapitel präsentierten Vorschläge sind ein unverzichtbarer Bestandteil des Forschungsprozesses, da sie direkte Einblicke in die Nutzererfahrungen und -präferenzen bieten. Durch die Analyse dieser Daten wird nicht nur ermittelt, welche Aspekte der aktuellen Lernwerkzeuge als effektiv oder ineffektiv angesehen werden, sondern es werden auch potenzielle Bereiche für zukünftige Verbesserungen identifiziert. Die Analyse der Verbesserungsvorschläge dazu, praxisnahe Empfehlungen für die Weiterentwicklung von SEE und anderen Lernwerkzeugen in der Softwareentwicklung zu formulieren.

5.5.2 *Ergebnisse*

Die Einteilung der Antworten in Themengebiete, hat folgende Auswertung gebracht:

- Visualisierung: 4x

- Steuerung: 3x
- Debugger: 3x
- Navigation: 2x
- Codeeditor: 2x
- Lesbarkeit: 1x
- Darstellungsformat: 1x
- Programmfluss: 1x

5.5.3 *Interpretation*

Die häufige Erwähnung der Visualisierung unterstreicht, dass die aktuelle Darstellung in der virtuellen Lernumgebung möglicherweise nicht optimal ist. Dies könnte auf eine unzureichende Klarheit oder Verständlichkeit der Visualisierung hindeuten, was das Empfinden negativ beeinflusst.

Die Betonung von Steuerung und Debugger lässt darauf schließen, dass diese Aspekte der Lernumgebung nicht intuitiv oder effizient genug gestaltet sind. Eine komplizierte Steuerung oder ein unzureichender Debugger können Frustration verursachen und das Verständnis der Probanden innerhalb des Projektes behindern.

Die Themen der Navigation und des Codeeditor könnten darauf hinweisen, dass die Navigation innerhalb des verfügbaren Debuggers nicht intuitiv ist, was Schwierigkeiten im Lernprozess verursacht.

Insgesamt legen die Ergebnisse nahe, dass die virtuelle Lernumgebung in ihrer jetzigen Form Mängel aufweist. Die identifizierten Themen sind Indikatoren für Bereiche, die einer dringenden Überarbeitung bedürfen, um die Lernumgebung effektiver und benutzerfreundlicher zu gestalten, damit der pädagogische Wert von SEE erhöht wird.

6

AUSBLICK

Für zukünftige Forschungen wäre es wertvoll, die Stichprobengröße zu erhöhen und eine diversere Gruppe von Probanden einzubeziehen, um die Generalisierbarkeit der Ergebnisse zu verbessern. Weiterhin wäre es sehr interessant neben eines zweistufigen Experiments, ein Experiment mit zwei Probandengruppen zu machen. So würden zwar subjektive Empfindungen nicht beide Umgebungen mit auffassen, allerdings würde dadurch sichergestellt, dass kein Lernprozess durch die erste Stufe existiert, der die zweite Stufe des Experiments beeinflusst.

Die Weiterentwicklung der SEE sollte die im Rahmen dieser Arbeit identifizierten Verbesserungsvorschläge berücksichtigen, insbesondere hinsichtlich der Benutzerfreundlichkeit und der intuitiveren Navigation. Zukünftige Forschungen könnten sich darauf konzentrieren, wie diese Anpassungen das Nutzererlebnis und das Programmverständnis beeinflussen.

Abschließend bietet diese Arbeit eine solide Grundlage für weiterführende Untersuchungen im Bereich der Softwarevisualisierung und des Programmverständnisses. Die gewonnenen Erkenntnisse und Vorschläge können dazu beitragen, die Entwicklung effektiverer und nutzerfreundlicherer virtueller Lernumgebungen in der Softwareentwicklung voranzutreiben.

7 FAZIT

7.1 LIMITATIONEN

Eine wesentliche Einschränkung dieser Arbeit liegt in der begrenzten Stichprobengröße und der spezifischen Auswahl der Probanden aus einem einzigen Unternehmen. Dies könnte zu einer Verzerrung der Ergebnisse geführt haben, da die Probanden möglicherweise ähnliche Arbeitsstile und Vorerfahrungen teilen. Zudem beschränkte sich die Untersuchung auf zwei spezifische Projekte, was die Generalisierbarkeit der Ergebnisse einschränken könnte.

Die Analysemethoden, insbesondere die qualitativen Ansätze, beinhalten subjektive Elemente in der Interpretation der Daten. Obwohl die Verwendung von Mayrings qualitativer Inhaltsanalyse einen strukturierten Rahmen bot, könnten unterschiedliche Interpretationen der Daten zu variierenden Schlussfolgerungen führen.

7.2 ABSCHLUSS

Diese Bachelorarbeit leistet einen Beitrag zum Verständnis der Rolle von Visualisierungstechniken im Hinblick auf die Verbesserung des Programmverständnisses gegenüber traditionellen Methoden. Durch die Gegenüberstellung von traditionellen Methoden und SEE wurden wertvolle Erkenntnisse über die Effektivität und Nutzerfreundlichkeit dieser Technologien gewonnen.

Die Ergebnisse zeigen, dass sowohl traditionelle als auch moderne Visualisierungstechniken effektiv das Verständnis von Softwarestrukturen und -abläufen fördern können. Allerdings weisen die unterschiedlichen Reaktionen und Bewertungen der Probanden darauf hin, dass jede Methode ihre spezifischen Stärken und Grenzen hat. Insbesondere SEE bietet einzigartige Möglichkeiten für ein tiefgreifendes Verständnis komplexer Programmstrukturen, benötigt jedoch weitere Optimierung.

Die Analyse der Berufserfahrung zeigte keinen signifikanten Einfluss auf die Antwortqualität, was darauf hindeutet, dass die verwendeten Lernmethoden unabhängig vom Erfahrungsniveau der Nutzer effektiv

sind.

Es werden ebenfalls Verbesserungsvorschläge für Lernwerkzeuge, insbesondere in Bezug auf Visualisierung, Steuerung und Navigation gegeben, welche wertvolle Ansatzpunkte für zukünftige Entwicklungen vor allem für SEE bieten.

A

GLOSSAR

Axivion Suite Die Axivion Suite ist ein umfassendes Software-Tool, das mehrere Schlüsselfunktionen besitzt, um die Qualität und Zuverlässigkeit von Softwareprojekten sicherzustellen 14

Code-City In der Code-City-Metapher werden Softwarekomponenten durch Gebäude in einer Stadt repräsentiert, wobei die Eigenschaften dieser Gebäude verschiedene Metriken der Software ausdrücken können — z. B. könnte die Höhe eines Gebäudes der Anzahl der Codezeilen entsprechen. 1, 10

Dark Theme Der "Dark Mode" (oder Dunkelmodus) ist eine Benutzeroberflächeneinstellung oder ein Design, das dunkle Farben für Hintergründe und helle Farben für Texte und andere Elemente verwendet. 11

Float Ein Datentyp zur Darstellung von Gleitkommazahlen 10

IntelliJ IDEA IntelliJ ist eine von JetBrains entwickelte Entwicklungsumgebung für mehrere Programmiersprachen. 11

Makefile Eine Makefile ist eine spezielle Datei, die häufig in der Softwareentwicklung verwendet wird, um den Build-Prozess eines Projekts zu automatisieren 14

B

AKRONYME

GXL standardisiertes Format zum Austausch von Graphdaten. Es wird häufig in der Softwaretechnik verwendet, insbesondere im Bereich der Softwarevisualisierung und des Reverse Engineering 13, 14

IDE IDE steht für „Integrated Development Environment“, was auf Deutsch „Integrierte Entwicklungsumgebung“ bedeutet. Innerhalb von Entwicklungsumgebungen schreiben Entwickler Software, da sie viele nützliche Funktionen beinhalten. 11

JLG JLG steht für Java-Log und beinhaltet Informationen zum Ablauf eines Java-Programms (vgl. [14]) 14

LOC LoC steht für „Line of Code“bzw. eine Zeile Quellcode 10

SEE Eine interaktive Visualisierung von Software, welche die *Code-City*-Metapher verwendet und einen kollaborativen Multiplayer über verschiedene Plattformen. 1, 3

UML Die UML (Unified Modeling Language) definiert eine allgemein verwendbare Modellierungssprache (auch Notation genannt) [13] 1

C

ANGEHÄNGTE DOKUMENTE

Bezeichnung	Beschreibung
Restaurant.zip	Das Erste der beiden Projekte, das im Experiment verwendet wird. Es enthält ein einfaches Programm in Java, welches ein Restaurant Szenario implementiert.
Werkstatt.zip	Das Zweite der beiden Projekte, das im Experiment verwendet wird. Es enthält ein einfaches Programm in Java, welches ein Werkstatt Szenario implementiert.
See.zip	Enthält die im Experiment verwendete kompilierte Version von SEE.
project1-material.pdf	Enthält die 8 PDF Seiten, die im Experiment verwendet wurden.
project2.gxl	Enthält die Graphischen Daten zu Projekt 2.
project2.jlg	Enthält die Programmabläufe zu Projekt 2.
BachelorExperimentResults-raw.csv	Enthält die Antworten der Probanden aus Microsoft Forms.
Project1-experience.xlsx	Enthält die Berufserfahrung der Probanden zu den Antworten von Projekt 1.
Project2-experience.xlsx	Enthält die Berufserfahrung der Probanden zu den Antworten von Projekt 2.
BachelorExperimentResults-feedback.csv	Enthält die Verbesserungsvorschläge der Probanden.
BachelorExperimentResults-likert.csv	Enthält die subjektiven Eindrücke der Probanden.
BachelorExperimentResults-Projekt-1-Korrektur.xlsx	Enthält die kategorisierten Antworten der Probanden zu Projekt 1.
BachelorExperimentResults-Projekt-2-Korrektur.xlsx	Enthält die kategorisierten Antworten der Probanden zu Projekt 2.

D

ABBILDUNGSVERZEICHNIS

LITERATURVERZEICHNIS

[1] Bill Albert and Tom Tullis. *Measuring the User Experience: Collecting, Analyzing, and Presenting UX Metrics*. Morgan Kaufmann, 2022.

[2] Earl R Babbie. *The practice of social research*. Cengage AU, 2020.

[3] John Biggs, Catherine Tang, and Gregor Kennedy. *Ebook: Teang for Quality Learning at University 5e*. McGraw-hill education (UK), 2022.

[4] Christoph Kecher. *UML 2 : das umfassende Handbuch*. Galileo Press, Bonn, 4., aktualisierte und erw. aufl. edition, 2011. Online Ressource (25311 KB, 448 S.).

[5] Robert F DeVellis and Carolyn T Thorpe. *Scale development: Theory and applications*. Sage publications, 2021.

[6] Norman Fairclough. *Language and power*. Routledge, 2013.

[7] Andy Field. *Discovering statistics using IBM SPSS statistics*. sage, 2013.

[8] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, 3 edition, 2004.

[9] Daniel Galin. *Software Quality Assurance: From Theory to Implementation*. Pearson Education, 2004.

[10] Joseph F Hair. Multivariate data analysis. 2009.

[11] John Hattie and Helen Timperley. The power of feedback. *Review of educational research*, 77(1):81–112, 2007.

[12] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.

[13] Christoph Kecher. *UML 2 - Das umfassende Handbuch*. Galileo Press, 2011.

[14] Lennart Kipka. Software-debugging mithilfe von code cities. 2022.

[15] Rainer Koschke. Three dimensional visualization of code changes in various parallel branches of software repositories in see. 2022.

[16] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage publications, 2018.

[17] Steven M. LaValle. *Virtual Reality*. Cambridge University Press, 2017.

[18] Philipp Mayring et al. Qualitative content analysis. *A companion to qualitative research*, 1(2):159–176, 2004.

[19] Mary L McHugh. The chi-square test of independence. *Biochemia medica*, 23(2):143–149, 2013.

[20] Michael Quinn Patton. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications, 2014.

[21] Philipp Mayring. *Qualitative Inhaltsanalyse : Grundlagen und Techniken*. Beltz Padagogik. Beltz, Weinheim, 12., uberarbeitete auflage edition, 2015. 1 Online-Ressource (152 Seiten) : Diagramme.

[22] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education, 2010.

[23] Robert W Service. Book review: Corbin, j., & strauss, a.(2008). basics of qualitative research: Techniques and procedures for developing grounded theory . thousand oaks, ca: Sage. *Organizational Research Methods*, 12(3):614–617, 2009.

[24] Steve McConnell. *Code complete*. Microsoft Press, 2nd ed. edition, 2004. xxxvii, 914.

[25] John Sweller. Cognitive load theory. In *Psychology of learning and motivation*, volume 55, pages 37–76. Elsevier, 2011.