Konzeption und Evaluation eines augmentierten Team-Raums zur Digitalisierung analoger Zeichenaktivitäten

Dissertation

zur Erlangung des akademischen Grades

Doctor Rerum Naturalium
(Dr. rer. nat.)

durch die
Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen, Campus Essen

vorgelegt von

Dipl.-Inform., Dipl.-Math. Markus Kleffmann
geboren am 07.07.1983 in Düsseldorf

Essen, 2018

Erstgutachter: Prof. Dr. Volker Gruhn
Zweitgutachter: Prof. Dr. Matthias Book
Tag der mündlichen Prüfung: 15.08.2018

Zusammenfassung

Die gemeinschaftliche Erstellung von informellen Freihandskizzen hat einen hohen Stellenwert im Software Engineering und in vielen anderen Design- und Ingenieursdisziplinen. Häufig werden hierfür traditionelle analoge Werkzeuge wie Whiteboards und Flip-Charts verwendet, die aufgrund ihres informellen Charakters und ihrer intuitiven Zugänglichkeit geschätzt werden. Die Verwendung rein analoger Zeichenwerkzeuge bringt jedoch auch zahlreiche Einschränkungen und Nachteile mit sich. In dieser Arbeit wird deshalb ein Konzept für einen technisch augmentierten Team-Raum vorgestellt, der diese Probleme durch den Einsatz elektronischer Whiteboards und mobiler Endgeräte beseitigt. Damit diese von Stakeholdern akzeptiert und intuitiv genutzt werden können, wurden spezielle Lösungen für verschiedene nicht-triviale Herausforderungen entwickelt:

Gezeichnete Linienzüge werden von einem Machine-Learning-System als Texte oder Formen klassifiziert und automatisch nach logischer Zusammengehörigkeit gruppiert, damit sie sinnvoll interpretiert und weiterverarbeitet werden können. Durch Anwendung heuristischer Tokenisierungsregeln werden die erkannten Texte in Worttupel zerlegt. Falls nötig werden diese Tupel geeignet aufbereitet, um potentiellen Handschrifterkennungsfehlern entgegenzuwirken. Darüber hinaus kommt ein retrospektiver Traceability-Algorithmus zum Einsatz, der zusammenhängende Inhalte miteinander verknüpft und Stakeholdern eine intuitive Navigation zwischen diesen ermöglicht. Die erstellten Freihandskizzen werden in einem Graphen verwaltet, der eine hierarchische Strukturierung sowie die gleichzeitige Darstellung mehrerer Abstraktionsebenen auf einem einzelnen elektronischen Whiteboard erlaubt. Basierend hierauf werden weitere Konzepte präsentiert, die Stakeholdern bei der Aufdeckung potentieller Inkonsistenzen, Widersprüche und Projektrisiken helfen können.

Der vorgestellte Ansatz wurde prototypisch implementiert und wird abschließend durch vier umfangreiche Experimente evaluiert.

Abstract

The collaborative creation of informal freehand sketches is very important in software engineering and many other design and engineering disciplines. It often uses traditional analog tools such as whiteboards and flipcharts, which are prized for their informal nature and intuitive accessibility. However, the use of purely analog drawing tools also brings numerous limitations and disadvantages. Therefore, this work presents a concept for a technically augmented team room, which eliminates these issues by using electronic whiteboards and mobile devices. In order to be accepted by stakeholders and used intuitively, special solutions had to be developed for various non-trivial challenges:

A machine-learning system classifies drawn strokes as texts or shapes and automatically groups them according to their logical affiliation so that they can be interpreted and processed properly. Recognized texts are split into word tuples by applying several heuristic tokenization rules. If necessary, these tuples are modified to counter potential handwriting recognition errors. In addition, a retrospective traceability algorithm is used to recover usable relationships between related content. The created freehand sketches are stored in a graph that allows hierarchical structuring and the simultaneous display of multiple abstraction levels on a single electronic whiteboard. Based on this, further concepts are developed that can help stakeholders to uncover potential inconsistencies, contradictions and project risks.

The presented approach was implemented prototypically and is finally evaluated by four extensive experiments.

Danksagung

An dieser Stelle möchte ich allen Personen danken, die mich bei meiner Dissertation unterstützt haben.

Mein besonderer Dank gilt meinem Doktorvater und dem Erstgutachter dieser Arbeit, Prof. Dr. Volker Gruhn, der mich stets mit seiner Erfahrung und seinem umfangreichen Wissen unterstützt hat. Ich hatte das große Glück, mich fünf Jahre lang intensiv und fast ausschließlich eigenen Forschungsfragen widmen zu dürfen. Für diese Freiheit und das mir entgegengebrachte Vertrauen bin ich sehr dankbar.

Ebenso möchte ich Prof. Dr. Matthias Book danken. Er fungiert als Zweitgutachter dieser Arbeit und hat mir während des Promotionsprozesses durch zahlreiche wertvolle Ratschläge und konstruktive Kritik enorm geholfen. Ohne seine engagierte Betreuung wäre die vorliegende Arbeit in dieser Form nicht möglich gewesen.

Darüber hinaus danke ich Sebastian Röhl für die tatkräftige Unterstützung bei der prototypischen Implementierung meines Konzepts. Bei Dr. Stefan Hanenberg möchte ich mich für die hilfreichen Hinweise zur statistischen Auswertung meiner Experimente und bei Nils Schwenzfeier für die Ratschläge zur Verwendung der logistischen Regression bedanken. Erik Hebisch und Dr. Simon Grapenthin danke ich für den fachlichen Austausch und die vielen spannenden Diskussionen zum Interaction Room. Weiterhin danke ich Marc Hesenius, der mir seine hybride Notation zur Beschreibung von Gesten für den AugIR-Prototyp zur Verfügung gestellt hat.

Ein besonders großer Dank gilt zudem meiner Familie, die mich durch den Promotionsprozess begleitet hat. Ich danke meiner Mutter Margret und meinen Schwiegereltern Sonja und Thomas für ihr Verständnis, dass ich abseits der Promotion so wenig Zeit hatte. Insbesondere möchte ich meiner Frau Nadja für ihre uneingeschränkte Unterstützung in den letzten Jahren danken. Sie hat mir stets den Rücken freigehalten und mir dadurch wesentlich dabei geholfen, dieses Projekt zu einem erfolgreichen Abschluss zu bringen.

Elisabeth Schäfer (1929 – 2017)
gewidmet.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Symbolverzeichnis

$\mathcal{A}$ Menge aller Zeichen im Unicode-Zeichensatz; wird als *Alphabet* bezeichnet

$\mathcal{A}^n$ Menge aller Zeichenketten der Länge n über $\mathcal{A}$

$\mathcal{A}^*$ Menge aller Zeichenketten beliebiger Länge über $\mathcal{A}$

$\mathcal{A}^{*^m}$ Menge aller Tupel mit m Zeichenketten beliebiger Länge über $\mathcal{A}$

$\mathcal{B}$ Menge aller an den x- und y-Achsen ausgerichteten Bounding Boxen (AABBs) in der euklidischen Ebene

β Leerzeichen; es ist $\beta \in \mathcal{S}$

ε Leere Zeichenkette der Länge 0

$\mathbb{N}$ Menge der natürlichen Zahlen

$\mathbb{N}_0$ Menge der natürlichen Zahlen einschließlich 0

$\mathcal{P}(M)$ Potenzmenge (d. h. Menge aller Teilmengen) einer Menge M

$\mathbb{R}$ Menge der reellen Zahlen

$\mathcal{S}$ Menge aller Sonderzeichen, definiert durch $\mathcal{S} := \mathcal{A} \setminus \mathcal{Z}$

$\mathcal{Z}$ Menge aller Standardzeichen, definiert durch $\mathcal{Z} := \{$„a“, . . . , „z“, „A“, . . . , „Z“, „0“, . . . , „9“, „ä“, „ö“, „ü“, „Ä“, „Ö“, „Ü“, „ß“$\} \subset \mathcal{A}$

Teil I.

Einführung und Problemstellung

1. Einleitung

Dieses Kapitel gibt eine Einführung in die vorliegende Arbeit, indem zunächst die Wichtigkeit von analogen Zeichenwerkzeugen wie Whiteboards und Flip-Charts für die Erstellung von informellen Skizzen im Software Engineering motiviert wird. Darauf aufbauend wird diskutiert, warum Team-Räume eine besondere Relevanz für die gemeinschaftliche Zusammenarbeit von Stakeholdern in heterogenen Softwareentwicklungsteams haben. Als Beispiel für einen solchen Team-Raum wird ausführlicher der sogenannte *Interaction Room* vorgestellt. Danach wird die Problemstellung der Arbeit erläutert, indem am Beispiel des Interaction Rooms diskutiert wird, welche Einschränkungen und Nachteile eine rein analoge Darstellung der Inhalte mit sich bringt und warum eine Verwendung von digitalen Whiteboards nützlich erscheint. Abschließend wird die zugrundeliegende Forschungshypothese vorgestellt, die im Rahmen dieser Arbeit untersucht wird. Das Kapitel schließt mit einem Ausblick auf die Gliederung der Arbeit und einem kurzen Überblick über die bisherigen Veröffentlichungen.

1.1. Informelles Skizzieren mit analogen Werkzeugen

Zahlreiche Studien zeigen die Wichtigkeit von Diagrammen und Skizzen im Software Engineering sowie in anderen Design- und Ingenieursdisziplinen [70, 82, 89, 110, 139, 204, 229, 235]. Sie können dabei helfen, Zusammenhänge besser zu verstehen, Probleme zu lösen, Ideen zu visualisieren und die Kommunikation zwischen beteiligten Stakeholdern anzuregen [43, 244, 259]. Darüber hinaus werden sie in der Softwareentwicklung häufig eingesetzt, um ein System aus verschiedenen Perspektiven und auf unterschiedlichen Abstraktionsebenen darzustellen [110]. Softwareingenieure

manifestieren dabei oftmals bereits in frühen Phasen eines Softwareprojekts ihr Systemverständnis in Diagrammen und Modellen [50]. Aus diesen Gründen wird deren Erstellung als wichtige und zentrale Tätigkeit in der Softwareentwicklung angesehen [18, 43, 56].

Hierbei wird für gewöhnlich zwischen Skizzen und Diagrammen unterschieden [36, 244]: Skizzen sind informeller Natur und werden häufig schnell ohne Berücksichtigung strenger formaler Richtlinien erstellt. Demgegenüber sind Diagramme formale grafische Repräsentationen von Modellen, die der strikten Syntax einer Modellierungssprache folgen.

Insbesondere Skizzen werden aufgrund ihres informellen Charakters und ihrer intuitiven Zugänglichkeit von professionellen Designern und Softwareingenieuren als effiziente Medien geschätzt [43, 140]. In einem typischen Softwareentwicklungsprojekt entstehen deshalb zahlreiche verschiedene Skizzen zur Veranschaulichung und Diskussion von Ideen [36, 43, 81, 185, 204, 234, 236]. Darüber hinaus werden sie häufig zum Brainstorming eingesetzt [228] und fördern die Zusammenarbeit von Stakeholdern [43, 259], was sie vor allem für kreative Tätigkeiten und Diskussionen wertvoll macht [81, 86, 137].

Weil die Entwicklung von Software eine Team-Aktivität ist, erfordert sie in der Regel die Zusammenarbeit mehrerer Personen [7, 98]. Die gemeinschaftliche Erstellung von Skizzen spielt deshalb in fast allen Softwareprojekten eine zentrale Rolle [39]. Persönliche und direkte Kommunikation sind hierbei die wichtigsten Faktoren für eine effiziente Zusammenarbeit [132].

Im Software Engineering werden häufig sogenannte computer-aided Software Engineering Tools (kurz CASE-Tools) zur Modellierung eingesetzt [134]. CASE-Tools unterstützen die Softwareingenieure bei der Erstellung grafischer Repräsentationen des Softwareprojekts und seiner Architektur in Form von Modellen und Diagrammen.

Klassische CASE-Tools haben jedoch einige nennenswerte Nachteile: Komplexe Diagramme werden häufig von einem Team aus Softwareingenieuren und nicht von einem einzelnen Stakeholder alleine entworfen [43, 156]. CASE-Tools sind jedoch nicht

darauf ausgelegt, von mehreren Personen gleichzeitig verwendet zu werden [6, 7]. Dies stellt eine Hürde für die gemeinschaftliche Zusammenarbeit dar.

Darüber hinaus verwenden CASE-Tools in der Regel formale Modellierungssprachen. Studien haben jedoch gezeigt, dass Softwareingenieure und Designer insbesondere in frühen Phasen eines Softwareprojekts freihändiges Zeichnen und informelle Notationen bevorzugen [43, 56, 140, 153]. Die meisten Modellierungssprachen wie beispielsweise UML [68] und BPMN [87] folgen jedoch einer sehr strikten Syntax. Eine solch strenge Formalisierung kann negative Auswirkungen auf die kreative Zusammenarbeit haben [211, 255]. Die meisten Stakeholder verwenden deshalb nur bestimmte Teile ausgewählter Modellierungssprachen oder greifen auf ihre eigene Notation zurück [43, 56, 186]. Darüber hinaus kann es förderlich sein, verschiedene Notationsformen miteinander zu kombinieren, während eine Restriktion auf eine bestimmte Notation oder Modellierungssprache schädlich sein und sich negativ auf den kreativen Austausch in einem Softwareprojekt auswirken kann [56, 81, 180, 259].

Deshalb werden CASE-Tools in der Praxis hauptsächlich für die Dokumentation ausgereifter oder finaler Modelle eingesetzt und die meisten Designer und Softwareingenieure verwenden analoge Whiteboards und Flip-Charts für die Durchführung ihrer Aktivitäten [39, 43, 47, 154, 185].

Die Arbeit mit diesen analogen Werkzeugen kann problemlos in einem Team erfolgen. Sie ermöglichen eine barrierefreie Interaktion, weil mehrere Stakeholder gleichzeitig auf einer entsprechend großen Fläche arbeiten und miteinander über das Gezeichnete diskutieren können [39]. Anstatt Stakeholder auf die Verwendung einer bestimmten Modellierungssprache einzuschränken, erlaubt die Verwendung analoger Zeichenwerkzeuge die Erstellung informeller und unvollständiger Skizzen. Hierbei können die Stakeholder beliebige Notationselemente verwenden, die sie für die jeweilige Situation als angemessen erachten. Die Arbeit mit analogen Zeichenwerkzeugen ist darüber hinaus intuitiv und erfordert keine speziellen Kenntnisse oder Fähigkeiten [50]. Deshalb werden Whiteboards und Flip-Charts häufig als effektivste Mittel für Diskussionen und Designaktivitäten angesehen [140].

1.2. Team-Räume und der Interaction Room

Die Entwicklung von Software ist eine Team-Aktivität [7, 98]. Sie schließt insbesondere die gemeinschaftliche Erstellung von Skizzen ein [43, 156] und erfordert hierzu eine kreative, soziale und interaktive Zusammenarbeit aller beteiligten Stakeholder in einem geeigneten Umfeld [239]. Hierbei sind persönlicher Kontakt und eine direkte Kommunikation besonders wichtig [19, 132, 179].

Aus diesem Grund beschäftigen sich zahlreiche Forschergruppen und Unternehmen schon seit vielen Jahren mit den Vorteilen sogenannter Team-Räume [231, 249]. Hierbei handelt es sich um physische Räume, die gelegentlich auch als Projekt-Räume [47] oder „War Rooms“ [163] bezeichnet werden. Stakeholder arbeiten dort zusammen an gemeinsamen Projekten. Studien haben gezeigt, dass dadurch sowohl die Produktivität als auch die persönliche Zufriedenheit der Teammitglieder steigt [47, 163, 231, 249].

Team-Räume können temporär Verwendung finden, wie beispielsweise für die Durchführung von Workshops, aber auch als dauerhafter Arbeitsplatz der Stakeholder für den gesamten Projektverlauf dienen [163]. In fast jedem Team-Raum werden dabei analoge Whiteboards und Flip-Charts aus den in Abschnitt 1.1 beschriebenen Gründen zur gemeinschaftlichen Erstellung von Skizzen, Diskussion von Ideen, Visualisierung von wichtigem Projektwissen und Beobachtung des Projektfortschritts verwendet [27, 43, 47, 156].

Projektteams, die komplexe Informationssysteme entwickeln, bestehen in der Regel aus einer Vielzahl unterschiedlicher Stakeholder, wie beispielsweise Fachexperten, Requirements Engineers, Systemarchitekten, Softwareentwicklern und Managern. Um ihre gemeinsamen Ziele zu erreichen, müssen sie ein einheitliches Problem- und Lösungsverständnis besitzen [29]. Dies ist jedoch für Mitglieder eines heterogenen Entwicklungsteams häufig schwierig, weil populäre Prozessmodelle wie beispielsweise Scrum [205] lediglich das organisatorische Rahmenwerk für den Entwicklungsprozess vorgeben aber keine effektive Unterstützung für die zielorientierte Auseinandersetzung mit den besonderen Herausforderungen eines Projekts bieten [27].

Um diesem Problem zu begegnen, wurde der *Interaction Room* [29] entwickelt: Hierbei handelt es sich um einen physischen Team-Raum, dessen Wände mit großen analogen Whiteboards ausgestattet sind. Stakeholder können im Verlauf von Workshops diese Zeichenflächen zur Erstellung und Diskussion diverser projektspezifischer Skizzen verwenden, um abstrakte Zusammenhänge komplexer IT-Projekte zu visualisieren und dadurch diskutierbar zu machen. Die Interaction-Room-Methode definiert hierzu eine Reihe sogenannter *Canvases*. Ein Canvas bezeichnet dabei eine bestimmte Perspektive auf das betrachtete Softwareprojekt. Jedem Whiteboard im Interaction Room wird explizit ein Canvas-Typ zugeordnet, der festlegt, welche Informationen von den Stakeholdern auf diesem Whiteboard skizziert und diskutiert werden sollten. Die folgenden Interaction-Room-Canvases werden im Rahmen dieser Arbeit betrachtet:

Feature Canvas Auf dem Feature Canvas werden die Anforderungen der Stakeholder an ein zu entwickelndes System gesammelt. Hierzu notieren die Stakeholder ihre Anforderungen in Form informeller User Stories auf Karteikarten und heften diese an den Feature Canvas, der in der Regel durch eine Metaplanwand oder ein Whiteboard repräsentiert wird. Die Karten werden zunächst in einer freien Brainstorming-Sitzung von den Teilnehmern in Einzelarbeit erstellt und anschließend gemeinsam diskutiert und priorisiert.

Process Canvas Ein Process Canvas visualisiert die Skizze eines Geschäftsprozesses, der durch das betrachtete IT-System realisiert wird. Das Ziel ist hierbei nicht die vollständige und präzise Spezifikation des Prozesses, sondern eine übersichtliche und verständliche Darstellung der zentralen Abläufe. Die verwendete Notation ist an UML-Aktivitätsdiagramme angelehnt: Prozessschritte werden durch beschriftete Rechtecke repräsentiert und durch Pfeile miteinander verbunden, die den Prozessfluss anzeigen.

Object Canvas Auf dem Object Canvas werden die wichtigsten fachlichen Geschäftsobjekte und Artefakte gesammelt, die in den auf den Process Canvases skizzierten Prozessen verarbeitet werden. Während Process Canvases somit die dynamischen Abläufe der Anwendungsdomäne darstellen, zeigt der Object Canvas die zentralen Entitäten und deren Beziehungen untereinander. Auch hier

liegt der Fokus auf einer intuitiven und verständlichen Darstellung, weshalb technische Details gewöhnlich nicht skizziert werden. Die verwendete Notation ist an UML-Klassendiagramme angelehnt: Objekte werden durch beschriftete Rechtecke repräsentiert, die zur Darstellung von Beziehungen durch Linien miteinander verbunden sind.

Integration Canvas Der Integration Canvas stellt das zu entwickelnde System im Kontext seiner umgebenden Systeme dar. Hierzu werden Schnittstellen und Abhängigkeiten zu anderen Komponenten der Systemlandschaft des Unternehmens betrachtet und visualisiert. Das zu entwickelnde System wird in der Mitte des Canvas platziert und relevante Umsysteme werden als beschriftete Rechtecke um dieses herum angeordnet. Pfeile zwischen den Entitäten stellen den Austausch von Daten- oder Informationsflüssen in Form von Objekten des Object Canvas dar.

Interaction Canvas Wenn ein zu entwickelndes System über komplexe grafische Dialoge und Benutzerschnittstellen verfügt, so können diese auf dem Interaction Canvas skizziert werden. Das Erscheinungsbild der relevanten Dialoge wird hierbei durch UI-Skizzen visualisiert und die Übergänge zwischen diesen werden in Form eines Storyboards beschrieben.

Abhängig von der Art des Projekts und des gegebenen Projektkontextes kommen in einem Interaction-Room-Workshop nicht immer alle Canvases zum Einsatz. In Abbildung 1.1 ist ein möglicher Aufbau eines Interaction Rooms mit vier Canvases dargestellt.

Obwohl der Ablauf eines Interaction-Room-Workshops flexibel gestaltet und an die jeweilige Projektsituation angepasst werden kann, definiert die Interaction-Room-Methode eine idealtypische Reihenfolge [29]: Üblicherweise werden zunächst informelle Anforderungen auf Karteikarten im Feature Canvas gesammelt. Die Karten werden anschließend von den Stakeholdern diskutiert, bewertet und priorisiert. Ausgewählte Anforderungen werden daraufhin als Prozessskizzen auf Process Canvases visualisiert. Wenn die Stakeholder hierbei relevante Geschäftsobjekte identifizieren, die in den jeweiligen Prozessen verarbeitet werden, so werden diese auf dem Object Canvas gesammelt und zueinander in Beziehung gesetzt. Nach Fertigstellung

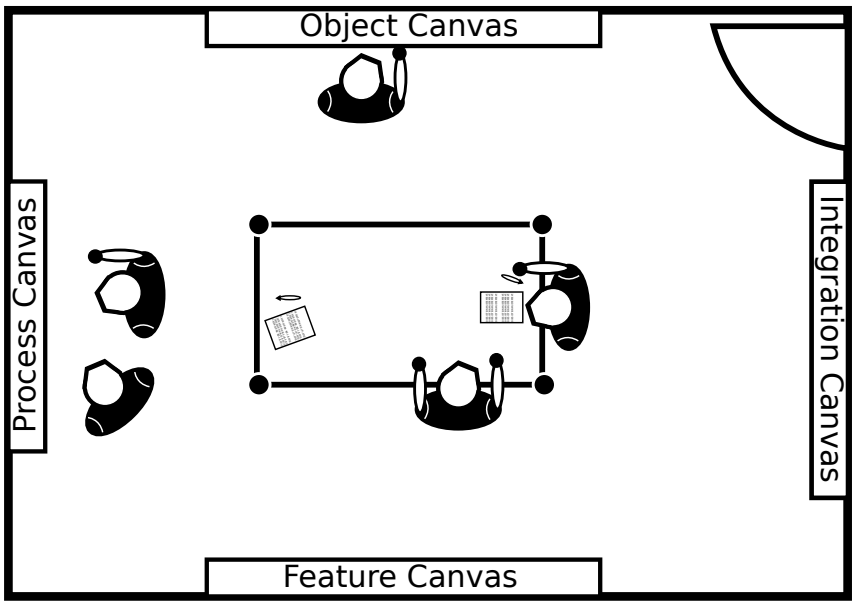

Abbildung 1.1.: Möglicher Aufbau eines Interaction Rooms mit Feature, Process, Object und Integration Canvas.

von Process und Object Canvases kann das zu entwickelnde System im Kontext seiner umliegenden Systeme betrachtet werden. Hierzu skizzieren die Stakeholder die entsprechenden Beziehungen auf dem Integration Canvas. Falls das zu entwickelnde System darüber hinaus ausreichend komplexe grafische Dialoge enthält, so kann zusätzlich auf dem Interaction Canvas ein Storyboard angefertigt werden, das die wichtigsten Dialoge skizziert und den Dialogfluss beschreibt.

Die auf den Canvases gemeinschaftlich erstellten Skizzen können mit sogenannten *Annotationen* versehen werden. Hierbei handelt es sich um Symbole mit einer festgelegten Bedeutung, die in Form von bedruckten Magneten oder Klebezetteln auf den Whiteboards platziert werden. Sie markieren besonders wichtige Teile des Systems und heben Aspekte innerhalb der Skizzen hervor, die nach Meinung der Stakeholder überdurchschnittlich komplex, ungewiss, aufwändig, kritisch oder generell entscheidend für den Projekterfolg sind. Dies erlaubt die Diskussion von wichtigen Wert-, Risiko- und Aufwandstreibern, die ansonsten häufig unerkannt bleiben und stärkt das gemeinsame Projektverständnis. Die Annotationen werden hierbei individuell von jedem Stakeholder ohne Abstimmung mit anderen Stakeholdern in den Skizzen

Tabelle 1.1.: Vier Beispiele für häufig verwendete Interaction-Room-Annotationen.

			
Sicherheit	Verbesserungsbedarf	Unveränderlichkeit	Ungewissheit

platziert und anschließend gemeinsam in der Gruppe diskutiert. Im Verlauf der Diskussion wird jede platzierte Annotation mit einer fortlaufenden Nummer versehen, um diese eindeutig einer zugehörigen Erläuterung zuzuordnen, die vom Schriftführer protokolliert wird.

Die Interaction-Room-Methode definiert insgesamt 21 verschiedene Annotationen [29]. Tabelle 1.1 zeigt beispielhaft die Symbole und deren Bedeutungen für vier häufig verwendete Annotationen:

- Die Annotation *Sicherheit* kennzeichnet Teile des Systems, in denen besondere Sicherheitsvorkehrungen getroffen werden müssen, die über die normalen Sicherheitsstandards des Unternehmens hinausgehen. Dies kann beispielsweise bei der Verarbeitung von sensiblen Nutzerdaten und Passwörtern der Fall sein.
- Die Annotation *Verbesserungsbedarf* kennzeichnet existierende Funktionen oder Prozesse, die in zukünftigen Revisionen angepasst, erweitert oder optimiert werden müssen. Hierbei kann es sich beispielsweise um einen Teil eines Geschäftsprozesses handeln, der aufgrund geänderter Anforderungen angepasst werden muss.
- Die Annotation *Unveränderlichkeit* kennzeichnet existierende Elemente des Systems, an denen zukünftig keine Änderungen mehr durchgeführt werden sollen. Dies kann zum Beispiel eine implementierte Funktionalität eines Alt-Systems betreffen, die zwar noch zuverlässig funktioniert, deren Implementierungsinterna jedoch soweit in Vergessenheit geraten sind, dass Aufwand und Risiko einer Anpassung oder Überarbeitung unverhältnismäßig hoch wären.
- Mit der Annotation *Ungewissheit* können Stakeholder Teile des Systems kennzeichnen, bei denen sie noch Klärungsbedarf sehen. Dies kann beispielsweise

fachliche Aspekte betreffen, die von einem Stakeholder noch nicht ausreichend verstanden wurden. Die Annotation kann darüber hinaus auch Fragen zur technischen Umsetzung sowie zu spezifischen Implementierungsdetails signalisieren.

1.3. Problemstellung

Die vorangegangenen Abschnitte machten deutlich, dass Stakeholder in Softwareprojekten häufig in einem geeigneten Umfeld wie einem Team- oder Projektraum zusammenarbeiten. Hierbei spielt die Verwendung von analogen Whiteboards und Flip-Charts oft eine zentrale Rolle. In Anlehnung an die Definition von Lipp und Will [149] werden Arbeitstreffen, bei denen sich mehrere Stakeholder gemeinsam einer zuvor definierten Aufgabe widmen, im Folgenden zusammenfassend als *Workshops* bezeichnet.

Die vorliegende Arbeit konzentriert sich dabei auf die Erstellung und Diskussion von Freihandskizzen im Rahmen solcher Workshops, weil dies eine wesentliche Aufgabe in der Softwareentwicklung darstellt [18, 43]. Hierzu soll zunächst untersucht werden, welche Einschränkungen und Probleme eine rein analoge Durchführung von Zeichenaktivitäten in Workshops mit sich bringt.

Jeder Workshop kann grundsätzlich grob in drei Phasen unterteilt werden [148, 149, 197]:

Vorbereitung In der Vorbereitungsphase eines Workshops wird dieser inhaltlich und organisatorisch geplant. Thema und Ablauf des Workshops werden festgelegt und es werden geeignete Stakeholder zur Teilnahme ausgewählt.

Durchführung Der idealtypische Ablauf eines Workshops richtet sich stark nach der betrachteten Thematik sowie der eingesetzten Workshop-Methodik. Insbesondere bei Workshops in Softwareprojekten sind jedoch die gemeinschaftliche Erstellung von Skizzen auf analogen Whiteboards sowie deren inhaltliche Diskussion fast immer wichtige Bestandteile der Durchführung.

Nachbereitung Im Anschluss an einen Workshop werden die skizzierten Inhalte und Notizen häufig analysiert und ausgewertet. Obwohl in dieser Phase keine gemeinschaftlichen Zeichenaktivitäten und Diskussionen mehr stattfinden, muss intensiv mit den erstellten Skizzen gearbeitet werden, um daraus weitere gewinnbringende Erkenntnisse abzuleiten und die Ergebnisse des Workshops nachvollziehbar zu dokumentieren.

Der Fokus dieser Arbeit liegt hauptsächlich auf der gemeinschaftlichen Erstellung von Skizzen, dem Umgang mit diesen sowie der Optimierung der hierzu verwendeten Werkzeuge. Weil diese Tätigkeiten in der Vorbereitungsphase eines Workshops im Allgemeinen keine Relevanz besitzen, wird diese Phase nicht näher betrachtet. Während für die Durchführung insbesondere die gemeinschaftliche Erstellung von Skizzen und die Arbeit mit diesen von Bedeutung sind, liegt das Hauptaugenmerk bei der Nachbereitung auf der Analyse und Auswertung der Inhalte. Beide Phasen werden deshalb nachfolgend genauer untersucht.

Auch im Interaction Room skizzieren Stakeholder gemeinschaftlich verschiedene Aspekte eines zu entwickelnden Systems auf Whiteboards. Die Erstellung und Diskussion dieser Skizzen ist ein zentraler Bestandteil jedes Interaction-Room-Workshops. Die im Folgenden identifizierten Probleme werden deshalb stets auf den Interaction Room übertragen, um sie an einer realen Workshop-Methode zu veranschaulichen. Obwohl die Nützlichkeit des Interaction Rooms bereits in verschiedenen Studien gezeigt wurde [26, 28, 84, 85], wird deutlich werden, dass die rein analoge Durchführung zahlreiche Einschränkungen und Nachteile mit sich bringt, die sich aus der Verwendung analoger Zeichenwerkzeuge ergeben.

1.3.1. Probleme bei der Workshop-Durchführung

In diesem Abschnitt werden Einschränkungen und Probleme diskutiert, die sich bei der analogen Durchführung von Workshops beobachten lassen.

Beschränkungen analoger Zeichenflächen

In Abschnitt 1.1 wurde dargelegt, dass analoge Whiteboards und Flip-Charts in der Praxis beliebte und intuitive Werkzeuge zur Erstellung informeller Skizzen sind. Obwohl sie den Stakeholdern größtmögliche Flexibilität und Freiheit bieten, bringen sie im Vergleich zu elektronischen Werkzeugen jedoch auch zahlreiche Einschränkungen mit sich [39, 50, 56]:

Weil analoge Zeichenwerkzeuge keinerlei Unterstützung für Zeichen- und Modellierungsaktivitäten bieten, können bereits triviale Änderungen an den Skizzen sehr aufwändig werden. Das Verschieben gezeichneter Elemente auf der Zeichenfläche kann beispielsweise unter Umständen das Ausradieren und Neuzeichnen großer Bereiche der entsprechenden Skizze erfordern, obwohl sich diese gegebenenfalls gar nicht verändert haben.

Darüber hinaus sind die Größen der physischen Zeichenflächen von Whiteboards und Flip-Charts sowie deren Anzahl in einem Team-Raum stark begrenzt. Obwohl es in bestimmten Situationen durchaus wünschenswert sein kann, dass die gemeinschaftlich erstellten Skizzen eine gewisse Komplexität nicht überschreiten, ist dies doch eine große Einschränkung. Sie kann dazu führen, dass Skizzen ausradiert und neu gemalt werden müssen, wenn im Verlauf eines Workshops neue Erkenntnisse gewonnen werden, die sich einer Zeichenfläche aus Platzmangel nicht mehr sinnvoll hinzufügen lassen.

Die vorgenannten Einschränkungen sind im Kontext des Interaction Rooms besonders problematisch, weil Layout und konkrete Inhalte der Skizzen a priori nicht feststehen sondern erst gemeinsam im Verlauf des Workshops von den Stakeholdern erarbeitet werden. Es ist deshalb zu Beginn der Erstellung einer Skizze häufig nur schwer abschätzbar, wie diese später aussehen wird.

Existenz von Medienbrüchen

Im Rahmen von Workshops werden häufig Informationen unterschiedlicher Art zusammengetragen, diskutiert und unter Verwendung verschiedener Medientypen er-

fasst. Dies schließt beispielsweise analoge Freihandzeichnungen auf Whiteboards, digitale Präsentationen, ergänzende Text-Dokumente sowie Aufzeichnungen der Workshop-Durchführung ein.

Auch in Interaction-Room-Workshops entstehen zahlreiche Artefakte in unterschiedlichen Medientypen. Wie in Abschnitt 1.2 erläutert, beginnt ein Interaction-Room-Workshop meist mit der Sammlung informeller Anforderungen auf dem Feature Canvas. Diese werden auf analogen Karteikarten notiert und an eine Metaplanwand geheftet. Die gesammelten Anforderungen werden gemeinschaftlich diskutiert, wobei die gewonnenen Erkenntnisse von einem Schriftführer in einem digitalen Textdokument erfasst werden.

Danach skizzieren die Stakeholder ausgewählte Anforderungen auf analogen Whiteboards in Process, Object und Integration Canvases, die darüber hinaus zusätzlich mit Magneten oder Klebezetteln annotiert werden können. Damit die erarbeiteten Ergebnisse nicht verloren gehen, müssen diese im Anschluss digitalisiert werden. Hierzu werden die Inhalte der Whiteboards in der Regel abfotografiert [50].

Wenn die Inhalte in nachfolgenden Workshops weiterverwendet werden sollen, müssen diese vorher geeignet aufbereitet werden. Aus den digitalen Fotos werden hierzu häufig analoge Poster erstellt, die den Stakeholdern in den nachfolgenden Workshops zur Verfügung stehen.

Darüber hinaus werden Interaction-Room-Workshops stets nachdokumentiert [29]. Hierzu werden die fotografierten handgezeichneten Skizzen in ordentliche Zeichnungen überführt, die mit einem CASE-Tool erstellt werden.

Die vorausgegangenen Ausführungen machen deutlich, dass bei der Durchführung und Nachbereitung von Workshops sowohl analoge als auch digitale Artefakte in unterschiedlichen Medientypen entstehen können. Es liegen somit oft zahlreiche Medienbrüche vor. Untersuchungen haben jedoch gezeigt, dass Medienbrüche in Softwareprojekten nach Möglichkeit vermieden werden sollten, weil sie zu einer Verfälschung der Daten führen, sich negativ auf deren Qualität auswirken und den gesamten Bearbeitungsprozess verlangsamen können [1, 72].

Unzureichende Verhaltensbeschreibung von grafischen Benutzerschnittstellen

Neben der Erstellung generischer Skizzen hat auch die Anfertigung von UI-Skizzen bei der Entwicklung interaktiver Anwendungen einen hohen Stellenwert [138]. Stakeholder erstellen hierbei bevorzugt handgezeichnete UI-Skizzen auf Whiteboards und Flip-Charts [14, 39, 43, 185]. Weil informell skizzierte Benutzerschnittstellen in der Regel mehr Feedback erhalten als formale Prototypen [109, 203, 255], sind sie insbesondere in frühen Phasen eines Softwareprojekts besser zur Generierung von Ideen geeignet [137, 193, 255]. Aus diesem Grund definiert beispielsweise die Interaction-Room-Methode den Interaction Canvas, der zur gemeinschaftlichen Erstellung eines Storyboards, zur informellen Skizzierung wesentlicher grafischer Benutzerschnittstellen sowie zur Beschreibung der Dialogflüsse verwendet werden kann.

Myers et al. [174] haben jedoch herausgefunden, dass die Beschreibung des dynamischen Verhaltens einer grafischen Benutzerschnittstelle häufig wesentlich schwieriger ist als die Vermittlung ihres optischen Erscheinungsbildes. Analoge UI-Skizzen auf Whiteboards und Flip-Charts vermitteln zwar intuitiv einen ersten Eindruck des Layouts, geben jedoch kein ausreichendes Gefühl für das Look and Feel einer Anwendung und ihrer relevanten Dialogübergänge [178]. Wenn diese Sachverhalte in einem Workshop thematisiert werden sollen, müssen die Stakeholder deshalb auf zusätzliche digitale Werkzeuge zur Erstellung von Mock-ups und Klick-Prototypen zurückgreifen.

Keine Unterstützung für vertikale und horizontale Fokuswechsel

Die Entwicklung komplexer Softwaresysteme manifestiert sich in der Regel in vielen logisch und physisch getrennten Artefakten [165]. Dies gilt auch für Interaction-Room-Workshops, in deren Verlauf wie zuvor beschrieben zahlreiche Informationen in Form unterschiedlicher Medientypen zusammengetragen werden. Eine effiziente Verwaltung dieser Artefakte ist eine bekannte Herausforderung in der Softwareentwicklung [5, 18, 61, 140].

In der Praxis ist es üblich, dass Stakeholder bei ihrer Arbeit mit Skizzen oft verschiedene nebeneinanderstellen und simultan betrachten [153, 185]. Darüber hinaus müssen sie häufig und regelmäßig den Fokus ihrer Betrachtung verschieben und zwischen logisch zusammenhängenden Artefakten wechseln, während sie an einer bestimmten Aufgabe arbeiten [140, 153, 174, 185, 212, 242, 261]. Dies kann beispielsweise notwendig sein, um alternative Designentscheidungen zu diskutieren [157], verschiedene Lösungsstrategien zu erproben [81] oder ein Problem aus unterschiedlichen Blickwinkeln zu betrachten [157].

Zum einen erfordern solche Fokuswechsel häufige Anpassungen des gewählten Abstraktionsniveaus [43, 81, 153, 156, 162, 185]. Es lässt sich beispielsweise beobachten, dass umfangreiche Diagramme in der Praxis zur Reduktion der Komplexität häufig in Unterdiagrammen verfeinert werden [170, 206]. Insbesondere komplexe Aktionen in großen Geschäftsprozessen werden oft in separaten Unterdiagrammen modelliert, um in dem übergeordneten Diagramm ein konsistentes Abstraktionsniveau beizubehalten. Stakeholder müssen bei ihrer Arbeit oft zwischen diesen Artefakten navigieren, um Probleme auf unterschiedlichen Abstraktionsebenen zu analysieren [185]. Sie benötigen deshalb eine einfache und intuitive Möglichkeit, das dargestellte Abstraktionsniveau zu wechseln. Hierbei erfordert die Bearbeitung von komplexen Aufgaben sowohl ein Verständnis der mikroskopischen als auch der makroskopischen Aspekte [43]. Es ist deshalb oft sinnvoll, den Stakeholdern gleichzeitig sowohl eine detaillierte Ansicht des Problems als auch eine verständliche Übersicht über den Problemkontext zur Verfügung zu stellen.

Darüber hinaus schließen die eingangs beschriebenen Fokuswechsel auch eine häufige Navigation zwischen zusammenhängenden Inhalten ein [81, 153, 174, 185]. Studien haben gezeigt, dass die Suche nach nützlichen Dokumenten, die Stakeholdern bei der Bewältigung ihrer aktuellen Aufgaben helfen könnten, ein bekanntes Problem in Softwareprojekten darstellt [3, 212]. Insbesondere in großen Projekten kann es schwierig sein, relevante Informationen zu identifizieren [31], weil diese oft nicht explizit miteinander verknüpft sind [5]. Somit existieren häufig viele potentiell nützliche Artefakte, deren Existenz den Stakeholdern eines Projekts gar nicht bewusst ist [88, 241]. Sie verbringen deshalb oftmals viel Zeit damit, relevante Informationen zu lokalisieren und den Zusammenhang zwischen diesen zu verstehen [130]. Lethbridge

et al. [142] haben sogar herausgefunden, dass die Suche nach nützlichen Inhalten oft dermaßen herausfordernd ist, dass viele Stakeholder dies mitunter gar nicht erst versuchen.

In Anlehnung an die von Sendall und Kozaczynski [206] gewählten Begrifflichkeiten, wird im Folgenden eine Anpassung des Abstraktionsniveaus als *vertikale Navigation* und ein Fokuswechsel zwischen logisch und inhaltlich zusammenhängenden Skizzen als *horizontale Navigation* bezeichnet. Beides muss so einfach und intuitiv wie möglich sein, damit Stakeholder schnell genug arbeiten können, um mit ihrem Gedankenprozess Schritt zu halten [70, 190]. Analoge Zeichenwerkzeuge wie Whiteboards und Flip-Charts bieten jedoch weder eine explizite Unterstützung für die adäquate Verwaltung und (hierarchische) Darstellung der Inhalte noch für eine effiziente vertikale und horizontale Navigation zwischen diesen.

Keine Unterstützung für Entwurf und Diskussion alternativer Lösungen

Studien haben gezeigt, dass Stakeholder häufig alternative Lösungen skizzieren und miteinander vergleichen [153]. Hierzu stellen sie oft verschiedene Versionen derselben Skizze gegenüber, die in Teilen identisch sind und sich nur in den zu diskutierenden Punkten voneinander unterscheiden. Eine Entwicklung solcher Lösungsalternativen ist jedoch mit analogen Werkzeugen nur schwer möglich, weil hierzu die Ausgangsskizze zunächst manuell vervielfältigt werden muss, bevor ausgehend davon verschiedene Varianten weiterentwickelt werden können [22].

Weiterhin wurde beobachtet, dass Stakeholder häufig zu älteren Ständen einer Skizze zurückkehren und experimentelle Ideen verwerfen, wenn sich diese nicht als sinnvoll erwiesen haben [153, 260]. Auf analogen Zeichenflächen ist es jedoch nicht möglich, Änderungen an einer Skizze zurückzunehmen. Sobald Elemente verändert oder ausradiert wurden, ist diese Anpassung dauerhaft und lässt sich nicht mehr umkehren. Auf diese Weise können falsche Entwurfsentscheidungen nicht einfach ungeschehen gemacht werden sondern erfordern unter Umständen das Neuzeichnen großer Teile einer Skizze. Insbesondere müssen alte Stände regelmäßig gesichert werden (bei-

spielsweise durch Abfotografieren), damit Stakeholder überhaupt zu diesen zurückkehren können.

Aufgrund der vorgenannten Einschränkungen sind Stakeholder bei der Verwendung analoger Zeichenwerkzeuge potentiell zurückhaltender hinsichtlich der Skizzierung experimenteller Ideen und alternativer Lösungen [153].

1.3.2. Probleme bei der Workshop-Nachbereitung

In diesem Abschnitt werden Einschränkungen und Probleme diskutiert, die sich bei der Nachbereitung und Auswertung analog durchgeführter Workshops beobachten lassen.

Aufwändige Persistierung der Inhalte

Skizzen und Diagramme, die von Stakeholdern gemeinsam erstellt wurden, geraten nach einem Workshop schnell in Vergessenheit [15]. Deshalb müssen deren Inhalte digitalisiert werden, um langfristig zur Verfügung zu stehen. Skizzen, die auf analogen Whiteboards und Flip-Charts gezeichnet wurden, werden hierzu in der Praxis normalerweise abfotografiert [18, 50]. Dies gilt auch für Interaction-Room-Workshops, für deren abschließende Dokumentation die fotografierten Inhalte in der Regel aufbereitet und mit einem CASE-Tool nachmodelliert werden.

Dieser manuelle Prozess hat jedoch einige Nachteile: Zum einen ist die digitale Rekonstruktion der Inhalte sehr zeitaufwändig – insbesondere für große Skizzen mit zahlreichen grafischen und textuellen Elementen und Annotationen. Zum anderen ist sie fehleranfällig, weil insbesondere bei komplexen Skizzen leicht Details übersehen und vergessen werden können.

Keine Sicherstellung der Konsistenz

Es ist ein bekanntes Problem im Software Engineering, dass Diagramme und Skizzen schnell inkonsistent werden können [185]. Je höher die Anzahl der erstellten Skizzen ist und über je mehr Abstraktionsebenen sich diese erstrecken, desto herausfordernder ist für gewöhnlich eine Wahrung der Konsistenz [93].

In Interaction-Room-Canvases können Inkonsistenzen nicht nur bei den skizzierten Inhalten sondern insbesondere auch bei den platzierten Annotationen auftreten. Wenn beispielsweise eine Aktion „Passwort verschlüsseln“ in mehreren Process Canvases vorkommt und in einem davon mit der Annotation „Sicherheit“ als sicherheitskritisch annotiert wird, dann trifft diese Charakterisierung meistens auch auf alle anderen Vorkommnisse des Elements in den anderen Canvases zu.

Die Interaction-Room-Methode schlägt deshalb vor, Annotationen eines Elements auf inhaltlich verknüpfte Elemente in anderen Canvases zu übertragen, um derartige Inkonsistenzen zu vermeiden [29]. Dies sollte jedoch nicht während eines Workshops sondern erst im Anschluss daran passieren, um die kognitive Leistung der Stakeholder durch fortlaufende Konsistenzprüfungen nicht negativ zu beeinflussen [29].

Analoge Zeichenwerkzeuge bieten allerdings keinerlei Unterstützung für die Identifikation inhaltlich zusammenhängender Elemente und die Übertragung von Annotationen zwischen diesen an. Aus diesem Grund ist eine entsprechende Wahrung der Konsistenz eine manuelle Tätigkeit, die abhängig von der Anzahl der erstellten Skizzen und der Menge der darin platzierten Annotationen sehr zeitaufwändig sein kann. Darüber hinaus ist sie fehleranfällig, weil bei einer manuellen Prüfung Zusammenhänge leicht übersehen werden können.

Keine Unterstützung für Skizzen-übergreifende Analysen

Bei der Nachbereitung eines Workshops können wichtige Informationen häufig aus einer Analyse der angefertigten Skizzen gewonnen werden. Hierzu reicht es allerdings oft nicht aus, jede Skizze für sich alleine zu betrachten. Bestimmte Aspekte

wie beispielsweise inhaltliche Inkonsistenzen und Widersprüche ergeben sich häufig nur aus einer Skizzen-übergreifenden Analyse, bei der mehrere Skizzen in einem gemeinsamen Kontext betrachtet werden.

Dies trifft auch auf die Nachbereitung von Interaction-Room-Workshops zu. Die Interaction-Room-Methode schlägt vor, die in den Canvases platzierten Annotationen im Anschluss an einen Workshop intensiv hinsichtlich potentieller Widersprüche und Projektrisiken zu analysieren [29].

Zum einen können platzierte Annotationen widersprüchlich sein: Wenn ein Element beispielsweise gleichzeitig mit den Annotationen „Unveränderlichkeit“ und „Verbesserungsbedarf“ gekennzeichnet ist, so impliziert dies einen potentiellen Widerspruch, weil eine gleichzeitige Bewahrung und Veränderung des Systems sich in der Regel gegenseitig ausschließen.

Zum anderen können platzierte Annotationen mögliche Projektrisiken implizieren: Wenn ein Element beispielsweise sowohl mit der Annotation „Sicherheit“ als auch mit der Annotation „Ungewissheit“ versehen wurde, deutet dies auf eine mögliche Gefahrenquelle hin, weil sicherheitsrelevante Aspekte eines Systems besonders gut durchdacht und von den Stakeholdern verstanden sein sollten.

Solche Widersprüche und Projektrisiken können leicht erkannt werden, wenn die Annotationen auf dem selben Element platziert wurden. Wenn jedoch das gleiche Element in mehreren Canvases vorkommt und an unterschiedlichen Stellen mit entsprechenden Annotationen versehen wurde, ist eine Identifikation der hierdurch eventuell entstehenden Widersprüche und Projektrisiken deutlich aufwändiger und schwieriger. Dies gilt insbesondere deshalb, weil analoge Zeichenwerkzeuge wie Whiteboards und Flip-Charts die Stakeholder nicht bei der Identifikation inhaltlich zusammenhängender Elemente unterstützen können.

1.3.3. Zusammenfassung

Insgesamt konnten bei der vorangegangenen Betrachtung der Durchführung und Nachbereitung von Workshops 12 nennenswerte Einschränkungen und Probleme

Tabelle 1.2.: Einschränkungen und Probleme bei der Verwendung analoger Zeichenwerkzeuge.

#	Beschreibung
1	Keine Unterstützung für Modifikation der Inhalte
2	Beschränkte Größe und Anzahl der Zeichenflächen
3	Existenz von Medienbrüchen
4	Unzureichende Verhaltensbeschreibung von Benutzerschnittstellen
5	Keine Unterstützung für Verwaltung der skizzierten Inhalte
6	Keine Unterstützung für vertikale Navigation
7	Keine Unterstützung für horizontale Navigation
8	Keine Unterstützung für Entwurf alternativer Lösungen
9	Aufwändige Persistierung der Inhalte
10	Keine Sicherstellung der Konsistenz
11	Keine Unterstützung für Identifikation von Widersprüchen
12	Keine Unterstützung für Identifikation von Projektrisiken

identifiziert werden, die sich aus der Verwendung rein analoger Zeichenwerkzeuge ergeben. Diese treffen insbesondere auch auf die Durchführung von Interaction-Room-Workshops zu. Tabelle 1.2 fasst diese noch einmal übersichtlich zusammen.

1.4. Lösungsansatz

Im vorherigen Abschnitt wurden Einschränkungen und Probleme bei der Verwendung analoger Zeichenwerkzeuge diskutiert. Die im Rahmen dieser Arbeit entwickelte Lösung basiert deshalb auf der Verwendung elektronischer Whiteboards. Diese erfreuen sich in Industrie und Forschung zunehmend größerer Beliebtheit [192, 227]. Zahlreiche Studien zeigen, dass die Verwendung großer (interaktiver) Displays grundsätzlich viele Vorteile mit sich bringt: Durch die große Darstellungsfläche können Bedeutung und Signifikanz bestimmter Informationen besser vermittelt werden, was Stakeholdern das Verständnis komplexer Systeme erleichtern kann [4]. Weiterhin ermöglichen entsprechend große Displays eine höhere Immersion beim Arbeiten [23] und erlauben es Stakeholdern, mehr Informationen auf einmal aufzunehmen [8]. Darüber hinaus steigert die physische Bewegung, die mit der Arbeit an großen

Bildschirmen einhergeht, oft die Leistung der Stakeholder und wird von ihnen in der Regel als sehr positiv wahrgenommen [16, 17]. Zudem bieten interaktive Whiteboards gegenüber ihren analogen Gegenstücken viele weitere allgemeine Vorteile, wie beispielsweise eine theoretisch unbegrenzte Anzahl von Zeichenflächen auf verhältnismäßig geringem Raum sowie das einfache Kopieren von Inhalten zur Skizzierung alternativer Ideen [156].

Zur Lösung der in Abschnitt 1.3 diskutierten Probleme bietet es sich daher an, die analogen Whiteboards und Flip-Charts durch elektronische Whiteboards zu ersetzen und zusätzlich mobile Endgeräte anzubinden. Hierdurch entsteht ein technisch augmentierter Team-Raum, der in Anlehnung an den Interaction Room im Folgenden als *Augmentierter Interaction Room* (kurz: *AugIR*) bezeichnet wird.

Die Verwendung digitaler Zeichenflächen verspricht hierbei eine sinnvolle Unterstützung der Stakeholder bei Zeichen- und Modellierungsaktivitäten. Medienbrüche lassen sich vollständig eliminieren, indem sämtliche Informationen bereits bei der Durchführung von Workshops in digitaler Form auf elektronischen Whiteboards und mobilen Endgeräten erfasst werden. Darüber hinaus können die Inhalte der Zeichenflächen automatisch analysiert und zusammengehörige Elemente miteinander verknüpft werden. Dies ermöglicht eine Unterstützung der Stakeholder bei der vertikalen und horizontalen Navigation. Weiterhin erlaubt eine Verknüpfung der Inhalte eine automatische Sicherstellung der Konsistenz sowie eine Identifikation potentieller Widersprüche und Projektrisiken.

Damit die digitalen Werkzeuge jedoch von den Stakeholdern als Ersatz für traditionelle analoge Whiteboards akzeptiert werden, müssen sich diese genau so intuitiv anfühlen und insbesondere auch für nicht-technische Stakeholder ebenso leicht zu bedienen sein. Dies erfordert die Konzeption spezieller Lösungen für mehrere nicht-triviale Probleme: Gezeichnete Inhalte und handschriftlich geschriebene Texte müssen erkannt und interpretiert werden können, um eine sinnvolle Weiterverarbeitung zu ermöglichen. Hierzu werden Algorithmen benötigt, die zuverlässig zwischen Texten und Formen unterscheiden sowie handgezeichnete Linienzüge entsprechend klassifizieren und nach logischer Zusammengehörigkeit gruppieren können. Die erkannten Texte müssen dabei geeignet aufbereitet werden, um beispielswei-

se potentiellen Handschrifterkennungsfehlern entgegenzuwirken. Zur Verknüpfung inhaltlich zusammenhängender Elemente muss überdies ein passender Traceability-Algorithmus entwickelt werden, der möglichst viele Zusammenhänge erfasst und gleichzeitig so wenig falsche Verknüpfungen wie möglich erzeugt. Basierend hierauf sind Konzepte zu erarbeiten, die Stakeholder bei der Aufdeckung von Inkonsistenzen, Widersprüchen und Projektrisiken unterstützen sowie eine intuitive Navigation zwischen zusammenhängenden Inhalten ermöglichen.

Entsprechende Lösungen für die vorgenannten Herausforderungen werden in Kapitel 3 erarbeitet, diskutiert und durch geeignete Experimente kalibriert. Ihre Praxistauglichkeit wird am Beispiel des Interaction Rooms im Rahmen mehrerer Studien in Teil III dieser Arbeit evaluiert.

1.5. Forschungshypothese

In diesem Abschnitt soll kurz die Forschungshypothese vorgestellt und erläutert werden, die dieser Arbeit zugrunde liegt. Sie lautet wie folgt:

Forschungshypothese

Bei der gemeinschaftlichen Arbeit mit Freihandskizzen entstehen Einschränkungen und Nachteile aus der Verwendung analoger Zeichenwerkzeuge. Durch eine technische Augmentierung der eingesetzten Werkzeuge können diese beseitigt werden. Am Beispiel der Interaction-Room-Methode lässt sich zeigen, dass sich hierbei keine negativen Auswirkungen für die Stakeholder ergeben.

Um die vorgenannte Forschungshypothese zu bestätigen, sind mehrere Schritte notwendig:

Zur grundlegenden Motivation müssen zunächst Einschränkungen und Nachteile identifiziert werden, die sich aus der Verwendung analoger Zeichenwerkzeuge ergeben. Dies erfolgte bereits in Abschnitt 1.3.

Darauf aufbauend ist ein detailliertes Konzept für einen Augmentierten Interaction Room zu erarbeiten und prototypisch zu implementieren, welches die in Abschnitt 1.3 aufgeführten Einschränkungen und Nachteile behebt. Das entsprechende Konzept wird in Kapitel 3 und die zugehörige technische Umsetzung in Kapitel 4 beschrieben.

Abschließend ist am Beispiel des Interaction Rooms zu zeigen, dass analoge Zeichenwerkzeuge grundsätzlich sinnvoll durch digitale Werkzeuge ersetzt werden können, ohne dass die technische Augmentierung hierbei negative Auswirkungen auf die Interaction-Room-Methode hat oder neue Probleme mit sich bringt. Hierzu erfolgt in Teil III dieser Arbeit im Rahmen mehrerer Studien ein direkter Vergleich zwischen analogen und augmentierten Interaction Rooms.

1.6. Gliederung der Arbeit

Dieser Abschnitt beschreibt kurz die Gliederung der vorliegenden Arbeit.

Kapitel 2 stellt relevante verwandte Forschungsarbeiten vor. Hierbei werden insbesondere Ansätze für technisch augmentierte Team-Räume, Werkzeuge zur Erstellung informeller Skizzen sowie Konzepte zur Klassifizierung von Linienzügen und zur Erfassung von inhaltlichen Zusammenhängen zwischen digitalen Artefakten betrachtet.

In Teil II dieser Arbeit wird eine Lösungsstrategie für die in Abschnitt 1.3 identifizierten Probleme entwickelt und eine prototypische Implementierung vorgestellt.

Kapitel 3 präsentiert hierzu im Detail das Konzept des Augmentierten Interaction Rooms. Es werden zunächst grundlegende Anforderungen an den AugIR sowie wichtige Definitionen und mathematische Grundlagen besprochen. Danach wird erläutert, wie in Freihandskizzen Texte von Formen unterschieden werden können, um eine automatische Handschrifterkennung auf den Inhalten zu ermöglichen. Basierend hierauf wird ein Algorithmus zur Erfassung von Zusammenhängen zwischen Freihandskizzen vorgestellt. Abschließend wird ausführlich diskutiert, wie hiermit

die in Abschnitt 1.3 beschriebenen Einschränkungen und Nachteile behoben werden können.

Kapitel 4 beschreibt die prototypische Implementierung des Augmentierten Interaction Rooms. Hierzu werden besondere Herausforderungen bei der technischen Umsetzung des Konzepts beleuchtet sowie relevante Details zu Architektur und Prozessen der AugIR-Software diskutiert.

Teil III dieser Arbeit präsentiert mehrere Studien, in denen die Praxistauglichkeit des entwickelten AugIR-Konzepts am Beispiel der Interaction-Room-Methode evaluiert wird.

Kapitel 5 betrachtet hierzu zunächst ganz allgemein die Erstellung und Annotierung von Skizzen auf elektronischen Whiteboards. Es wird untersucht, ob diese genau so einfach und intuitiv zu verwenden sind wie traditionelle analoge Whiteboards und inwiefern sie Stakeholder bei der Arbeit mit Freihandskizzen unterstützten können.

In Kapitel 6 wird das digitale Storyboard des AugIRs evaluiert. Es wird untersucht, ob es Stakeholdern das Verständnis komplexer Gesten erleichtert und wie gut sie hierauf Dialogflüsse spezifizieren können.

Kapitel 7 analysiert die Qualität der Verknüpfungen, die der AugIR automatisch zwischen zusammenhängenden Elementen erfasst. Es wird überprüft, wie viele Zusammenhänge durchschnittlich korrekt ermittelt werden können und wie viele falsche Beziehungen der hierzu entwickelte Algorithmus vorschlägt.

Die Evaluation des AugIRs wird mit einer Betrachtung der Verwendung der automatisch erfassten Verknüpfungen in Kapitel 8 abgeschlossen. Hierbei wird untersucht, wie gut diese in der Praxis die Stakeholder tatsächlich bei der Navigation zwischen Skizzen sowie bei der Identifikation von inhaltlichen Zusammenhängen unterstützen können.

Teil IV stellt den Schlussteil dieser Arbeit dar.

Hierbei fasst Kapitel 9 noch einmal die wesentlichen Beiträge der vorliegenden Arbeit zusammen und diskutiert kritisch das vorgestellte Lösungskonzept des Augmentier-

ten Interaction Rooms im Hinblick auf die eingangs formulierte Forschungshypothese.

Kapitel 10 schließt die Arbeit mit einem Ausblick auf mögliche zukünftige Forschungstätigkeiten und Erweiterungen des AugIR-Konzepts ab.

1.7. Eigene Veröffentlichungen

In diesem Abschnitt werden kurz die bisherigen Publikationen zum Augmentierten Interaction Room vorgestellt. Tabelle 1.3 zeigt eine Übersicht über alle Veröffentlichungen im Zeitraum 2013 bis 2017.

Tabelle 1.3.: Eigene Publikationen zum Augmentierten Interaction Room.

Jahr	Titel	Autor(en)	Konferenz / Journal
2017	Evaluation of a Traceability Approach for Informal Freehand Sketches	Markus Kleffmann, Sebastian Röhl, Matthias Book, Volker Gruhn	International Journal on Automated Software Engineering
2016	Sketching Gesture-Based Applications in a Collaborative Working Environment with Wall-Sized Displays	Marc Hesenius, Markus Kleffmann, Volker Gruhn	International Conference on Collaboration Technologies and Systems (CTS)

Tabelle 1.3.: (Fortsetzung)

Jahr	Titel	Autor(en)	Konferenz / Journal
2015	Establishing and Navigating Trace Links Between Elements of Informal Diagram Sketches	Markus Kleffmann, Sebastian Röhl, Volker Gruhn, Matthias Book	International Symposium on Software and Systems Traceability (SST)
2015	Connecting UI and Business Processes in a Collaborative Sketching Environment	Markus Kleffmann, Marc Hesenius, Volker Gruhn	ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS)
2014	Supporting Collaboration of Heterogeneous Teams in an Augmented Team Room	Markus Kleffmann, Matthias Book, Volker Gruhn	International Workshop on Social Software Engineering (SSE)
2014	Navigation among Model Sketches on Large Interactive Displays	Markus Kleffmann, Matthias Book, Volker Gruhn	IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW)

Tabelle 1.3.: (Fortsetzung)

Jahr	Titel	Autor(en)	Konferenz / Journal
2014	Automated Versioning and Temporal Navigation for Model Sketches on Large Interactive Displays	Markus Kleffmann, Matthias Book, Erik Hebisch, Volker Gruhn	ACM Symposium on Applied Computing (SAC)
2014	AugIR – The Conceptual Design and Evaluation of an Augmented Interaction Room	Markus Kleffmann	ACM/IEEE International Conference on Automated Software Engineering (ASE)
2013	Towards Recovering and Maintaining Trace Links for Model Sketches across Interactive Displays	Markus Kleffmann, Matthias Book, Volker Gruhn	International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)

Die umfangreiche Journal-Veröffentlichung „Evaluation of a Traceability Approach for Informal Freehand Sketches“ [128] präsentiert die in den Abschnitten 3.6, 3.7 und 3.8 diskutierten Lösungsansätze zur Tokenisierung von Texten und zur Erfassung von Zusammenhängen zwischen Interaction-Room-Skizzen. Darüber hinaus werden die in Kapitel 7 beschriebenen Ergebnisse der zugehörigen Studie vorgestellt.

In dem Konferenz-Papier „Sketching Gesture-Based Applications in a Collaborative Working Environment with Wall-Sized Displays“ [102] wird das Konzept des digitalen Interaction Canvases vorgestellt, der in Abschnitt 3.9.2 beschrieben wird.

Darüber hinaus präsentiert das Paper die Ergebnisse einer Vorstudie zur Evaluation dieses Ansatzes.

Die Veröffentlichung „Establishing and Navigating Trace Links Between Elements of Informal Diagram Sketches“ [123] beschreibt eine erste Version des in Abschnitt 3.8 vorgestellten Verfahrens zur Erfassung von Zusammenhängen zwischen Canvases. Darüber hinaus werden mögliche Anwendungszwecke für die erstellten Verknüpfungen diskutiert, wie beispielsweise die in Abschnitt 3.12.1 vorgestellte Übertragung von Annotationen zwischen verbundenen Elementen.

In der Publikation „Connecting UI and Business Processes in a Collaborative Sketching Environment“ [127] wird motiviert, wieso eine Erfassung von Zusammenhängen zwischen informellen Skizzen im Interaction Room nützlich sein kann. Weiterhin wird der in Abschnitt 3.11.2 vorgestellte Ansatz zu Visualisierung dieser Verknüpfungen besprochen.

Das Workshop-Papier „Supporting Collaboration of Heterogeneous Teams in an Augmented Team Room“ [125] fokussiert auf die zwischenmenschlichen Aspekte der Zusammenarbeit zwischen Stakeholdern in Interaction-Room-Projekten. Im Fokus steht hierbei das in Abschnitt 3.12.1 vorgestellte Konzept zur Übertragung von Annotationen auf verbundene Elemente.

Die Veröffentlichung „Navigation among Model Sketches on Large Interactive Displays“ [122] befasst sich insbesondere mit der in Abschnitt 3.10 beschriebenen hierarchischen Verwaltung und Darstellung der Inhalte als Graph sowie mit den in Abschnitt 3.11 beschriebenen Navigationstechniken zwischen Skizzen.

Das Konferenz-Papier „Automated Versioning and Temporal Navigation for Model Sketches on Large Interactive Displays“ [126] motiviert die Nützlichkeit einer automatischen Erfassung von Änderungen in Interaction-Room-Skizzen sowie die Möglichkeit, zwischen diesen Änderungen zu navigieren.

Die Veröffentlichung „AugIR – The Conceptual Design and Evaluation of an Augmented Interaction Room“ [124] gibt einen Überblick über das Gesamtkonzept des Augmentierten Interaction Rooms, der in Kapitel 3 vorgestellt wird. Es handelt

sich um einen Beitrag für das Doctoral Symposium der ACM/IEEE International Conference on Automated Software Engineering.

Die Publikation „Towards Recovering and Maintaining Trace Links for Model Sketches across Interactive Displays“ [121] ist die erste jemals erschienene Veröffentlichung zum Augmentierten Interaction Room. Sie beschreibt die Grundidee des AugIRs und fokussiert dabei insbesondere auf das in Abschnitt 3.8 vorgestellte Verfahren zur Erfassung von Zusammenhängen zwischen Skizzen.

2. Verwandte Arbeiten und Stand der Forschung

Dieses Kapitel diskutiert relevante verwandte Ansätze und gibt einen Überblick über den aktuellen Stand der Forschung. Zunächst werden kurz einige allgemeine Arbeiten zu technisch augmentierten Team-Räumen vorgestellt. Danach werden Werkzeuge präsentiert, die in solchen Räumen zur gemeinschaftlichen Erstellung von informellen Freihandskizzen zum Einsatz kommen. Darüber hinaus werden Ansätze zur Klassifizierung von Linienzügen sowie zur Verknüpfung inhaltlich zusammenhängender Inhalte diskutiert, bevor das Kapitel mit einer Bewertung der verwandten Arbeiten schließt.

2.1. Augmentierte Team-Räume

In Abschnitt 1.2 wurde bereits die Relevanz von Team-Räumen für die gemeinschaftliche Zusammenarbeit in Softwareentwicklungsprojekten diskutiert. Es existieren zahlreiche Ansätze, die diese Räume durch den Einsatz von Technologie erweitern.

Stefik et al. [219, 220] haben bereits in den 80er Jahren erkannt, dass ein Einsatz von großen interaktiven Displays in Team-Räumen nützlich sein kann. Sie stellen mit *CoLab* einen Arbeitsraum vor, der jedem Teammitglied einen Platz mit eigenem Computer zur Verfügung stellt. Die PCs sind miteinander über ein Netzwerk verbunden und darüber hinaus an einen großen berührungssensitiven Bildschirm

angeschlossen. Weiterhin verfügt der Raum über ein Stehpult mit einem Keyboard, das als dedizierter Arbeitsplatz von einem Moderator genutzt werden kann.

Auch Pederson et al. [184] verwenden ein elektronisches Whiteboard zur technischen Augmentierung von Team-Räumen. Sie stellen hierzu *Tivoli* vor, welches als früher Vorreiter für die im nächsten Abschnitt diskutierten Werkzeuge zur Erstellung digitaler Freihandskizzen gilt. *Tivoli* unterstützte bereits damals die gleichzeitige Bearbeitung von Skizzen durch mehrere Nutzer auf dem gleichen Whiteboard sowie die Verwendung einfacher Gesten.

Geyer et al. [80] präsentieren mit *IdeaVis* einen alternativen Ansatz für einen augmentierten Arbeitsraum, der die gemeinschaftliche Erstellung von Papierskizzen durch Verwendung digitaler Displays ergänzt. Als Hauptarbeitsfläche kommt in *IdeaVis* ein gewöhnlicher analoger Tisch zum Einsatz, auf dem Stakeholder traditionell mit Stift und Papier arbeiten. Die verwendeten Stifte sind mit Miniaturkameras ausgestattet und das Papier ist mit einem speziellen Punktmuster bedruckt. Auf diese Weise können analog gezeichnete Linienzüge digital erfasst und über eine Bluetooth-Verbindung zur Darstellung an einen großen Bildschirm gesendet werden. Das Papier ist darüber hinaus mit Symbolen versehen, welche die direkte Auslösung bestimmter Aktionen ermöglichen, wie beispielsweise die Ausgabe der gezeichneten Inhalte auf einem Drucker. Ein sehr ähnlicher Ansatz wird auch von Brandl et al. [32] vorgeschlagen.

Statt elektronischen Whiteboards kommen in augmentierten Team-Räumen häufig auch digitale Tischplatten (engl. Tabletops) zum Einsatz. Shen et al. [209] präsentieren mit *UbiTable* einen Ansatz, der Benutzern den einfachen Austausch und die gemeinsame Betrachtung von Dokumenten ermöglicht. Hierzu verbindet sich der digitale Tisch automatisch mit dem Laptop eines Nutzers und ermöglicht einen nahtlosen Transfer beliebiger Artefakte zwischen beiden Geräten. Dokumente können auf dem Display des Tisches in einem gemeinsam genutzten Bereich frei angeordnet und von den Beteiligten manipuliert werden.

Viele Ansätze kombinieren auch die Verwendung beider Gerätetypen. Hilliges et al. [105] stellen beispielsweise einen Raum vor, in dem gleichzeitig sowohl ein interaktives Whiteboard als auch ein Tabletop zum Einsatz kommen. Nutzer können auf

dem digitalen Tisch informelle Ideen in Form digitaler Notizzettel skizzieren. Diese können zur gemeinschaftlichen Diskussion auf das verbundene elektronische Whiteboard übertragen und dort frei angeordnet, gruppiert und manuell miteinander verknüpft werden.

Insbesondere neuere Ansätze nutzen darüber hinaus auch die Möglichkeiten, die sich durch den Einsatz mobiler Geräte ergeben. Hailpern et al. [94] präsentieren mit *TEAM STORM* beispielsweise ein System, das bis zu fünf mobile Endgeräte mit einem großen interaktiven Display verbindet. Nutzer können auf ihren individuellen Tablets in einem privaten Workspace arbeiten und die skizzierten Inhalte über das gemeinsam genutzte große Display miteinander teilen und diskutieren.

Weitere Arbeiten zeigen, dass darüber hinaus auch andere Elemente eines Team-Raums sinnvoll durch verschiedenartige elektronische Komponenten ersetzt werden können.

Streitz et al. [221, 223] stellen mit *Roomware* beispielsweise einen augmentierten Team-Raum vor, der zusätzliche Informations- und Kommunikationstechnologie in unterschiedlichen Raumelementen wie Stühlen, Tischen und Wänden platziert. Durch eine Ausstattung der Umgebung mit verschiedenen Sensoren und Netzwerkschnittstellen bietet *Roomware* neue Möglichkeiten zur Interaktion mit Daten und Informationen, indem es diese allgegenwärtig verfügbar macht. Nutzer können digitale Artefakte auf unterschiedlichen Geräten wie Tabletops, elektronischen Whiteboards, mobilen Endgeräten und berührungssensitiven Stuhlarmlehnen darstellen, manipulieren und austauschen.

Ein weiterer Ansatz wird von Haller et al. [95] präsentiert. Sie stellen mit dem *NiCE Discussion Room* einen augmentierten Team-Raum vor, der zur Unterstützung von Gruppendiskussionen ebenfalls verschiedenartige augmentierte Elemente enthält. Nutzer können ein großes elektronisches Whiteboard zur gemeinschaftlichen Erstellung von Skizzen sowie zur Darstellung und Diskussion anderer digitaler Artefakte verwenden. Darüber hinaus können sie – ähnlich wie bei *IdeaVis* – analoge Skizzen mit speziellen Stiften erstellen und diese automatisch digitalisieren lassen. Weiterhin lassen sich mobile Geräte wie Laptops und Tablets in die Umgebung

integrieren, um einen nahtlosen Austausch von Daten und Informationen zu ermöglichen.

Weitere Beispiele für augmentierte Team-Räume, in denen insbesondere große elektronische Whiteboards und/oder Tabletops zum Einsatz kommen, sind *Dolphin* [222], *i-Land* [224, 230], *Caretta* [226], *Dynamo* [113], *TATIN-PIC* [115] und *EnhancedTable* [131].

2.2. Informelles Skizzieren

Wie in Abschnitt 1.1 erläutert hat die gemeinschaftliche Erstellung informeller Freihandskizzen im Software Engineering einen hohen Stellenwert. Dabei ist sowohl die Anfertigung generischer Skizzen als auch die Erstellung von UI-Skizzen von großer Bedeutung [138, 256]. Zahlreiche Forschungsarbeiten beschäftigen sich mit den allgemeinen Vorteilen digitaler Zeichenwerkzeuge [14, 90, 114, 177, 188, 220], der Erstellung informeller Freihandskizzen [14, 37, 40, 45, 49, 50, 89, 97, 107, 116, 129, 138, 146, 150, 156, 164, 176, 177, 178, 188, 189, 218, 257], der Erkennung und Interpretation der Inhalte [96, 100, 137, 183], der Verwaltung von Skizzen [22, 80, 90, 129, 176, 175, 177, 220, 222] sowie dem räumlich verteilten gemeinschaftlichen Skizzieren [92, 94, 199].

Dieser Abschnitt gibt einen Überblick über relevante verwandte Forschungsarbeiten zur Erstellung digitaler Freihandskizzen und beleuchtet darüber hinaus insbesondere Ansätze zur effizienten Verwaltung der erstellten Artefakte, weil dies eine bekannte Herausforderung in der Softwareentwicklung darstellt [18]. Diese Aspekte sind auch für den AugIR von besonderer Bedeutung, weil die Erstellung und Verwaltung von handschriftlichen Skizzen ein wesentlicher Bestandteil jedes Interaction-Room-Workshops ist.

2.2.1. Erstellung von digitalen generischen Skizzen

Mangano et al. [155, 156] präsentieren mit *Calico* ein Werkzeug zur Erstellung informeller Skizzen in den frühen Phasen eines Softwareentwicklungsprojekts. Es ermöglicht Stakeholdern die Anfertigung von Freihandskizzen auf elektronischen Whiteboards und Tablets. Um informell die besondere Wichtigkeit bestimmter Bereiche einer Skizze zu betonen, können diese in sogenannte „Scraps“ umgewandelt werden. Diese entsprechend markierten Bereiche werden besonders hervorgehoben und können frei auf der Zeichenfläche verschoben und miteinander kombiniert werden. *Calico* stellt dabei verschiedene Arten von Scraps für unterschiedliche Anwendungszwecke zur Verfügung: Während beispielsweise „List Scraps“ die entsprechenden Inhalte in Form vertikaler Auflistungen anordnen, ermöglichen „Text-Scraps“ eine Texteingabe über ein angeschlossenes Keyboard. Die gemeinschaftliche Erstellung von Skizzen auf mehreren Geräten wird unterstützt, indem Stakeholder synchron auf der gleichen Zeichenfläche sowie asynchron auf verschiedenen Zeichenflächen arbeiten können. Darüber hinaus bietet *Calico* diverse Komfort-Funktionen zur Bearbeitung digitaler Skizzen, wie beispielsweise das Klonen existierender Inhalte und die manuelle Verknüpfungen logisch zusammenhängender Skizzen über sogenannte „Tags“.

Loksa et al. [150] haben *Calico* in einem Hochschulkurs für Software Design eingesetzt und untersucht, inwiefern Sketching-Tools Studenten bei ihrer Arbeit im universitären Umfeld unterstützen können. Sie fanden heraus, dass die Verwendung entsprechender Werkzeuge Studenten bei der gemeinschaftlichen Lösung von Design-Problemen hilft und eine schnellere Entwicklung, Verfeinerung und Evaluation von Ideen ermöglicht.

Stapleton et al. [218] kombinieren traditionelle Keyboard- und Maus-Interaktionen mit Stifteingaben. Sie präsentieren ein Tool, das die Erstellung von Diagrammen und Skizzen in zwei verschiedenen Modi ermöglicht: Zum einen können Stakeholder Diagramme mit einem formalen Diagrammeditor erstellen, dessen Interface sich an klassischen CASE-Tools orientiert. Zum anderen können sie informelle Skizzen mit einem Stift zeichnen. Stakeholder können frei zwischen diesen beiden Modi wechseln.

Um die Konsistenz der Inhalte zu erhalten, können die erstellten Freihandskizzen darüber hinaus in formale Diagramme konvertiert werden und umgekehrt.

Ein ähnlicher Ansatz wurde bereits zuvor von Hammond und Davis [97] mit *Tahuti* verfolgt: Die Autoren wollen die Einfachheit eines informellen Zeichenwerkzeugs mit der Modellierungsunterstützung eines UML-Editors kombinieren. *Tahuti* ermöglicht hierzu die Erstellung von Freihandzeichnungen auf großen Multi-Touch-Bildschirmen. Stakeholder können dabei entscheiden, ob gezeichnete Elemente so dargestellt werden sollen, wie sie gemalt wurden, oder ob sie als UML-Elemente interpretiert und durch entsprechende Symbole ersetzt werden sollen. Zur Interpretation der Inhalte wird eine mehrstufige Mustererkennung verwendet.

Wüest et al. [257] präsentieren mit *FlexiSketch* eine mobile Anwendung zur Erstellung von Freihandskizzen. Sie motivieren, dass Softwareingenieure ihre Modellierungsaktivitäten insbesondere dann mit Stift und Papier durchführen, wenn sie im Außeneinsatz sind, beispielsweise um Informationen und Anforderungen von Stakeholdern zu gewinnen. Deshalb wurde *FlexiSketch* speziell für den Einsatz auf mobilen Endgeräten konzipiert. Benutzer können auf diesen informelle Skizzen erstellen, sie mit Annotationen anreichern und später zur Weiterverarbeitung in semiformale Diagramme transformieren.

In einer neueren Arbeit stellen Wüest et al. [258] mit *FlexiSketch Team* eine Erweiterung ihres ursprünglichen Ansatzes vor. *FlexiSketch Team* ermöglicht insbesondere die gemeinschaftliche Erstellung von Skizzen durch Verwendung mehrerer Tablets und eines elektronisches Whiteboards, wodurch mehrere Stakeholder gleichzeitig auf verschiedenen Geräten an denselben Skizzen arbeiten können. Darüber hinaus stellen sie eine Desktop-Anwendung zur Verfügung, die eine gemeinsame Zeichenfläche für alle Stakeholder auf einem elektronischen Whiteboard realisiert.

Chen, Grundy und Hosking [40] präsentieren *SUMLOW* zur Erstellung von Freihandskizzen auf elektronischen Whiteboards. Durch Anwendung von Handschrift- und Mustererkennung können die erstellten Skizzen in formale UML-Diagramme transformiert und zur weiteren Verwendung in externen CASE-Tools exportiert werden. *SUMLOW* unterstützt dabei insbesondere die Erkennung und Formalisierung

von Klassen-, Sequenz- und Use-Case-Diagrammen. Basierend auf dieser Arbeit haben Grundy und Hosking [89] *Marama* entwickelt. Dieses Tool ermöglicht die Erstellung domänenspezifischer grafischer Modellierungswerkzeuge als Plug-Ins für die populäre Entwicklungsumgebung Eclipse.

Ju et al. [116] präsentieren mit *Range* eine Anwendung für interaktive Whiteboards, die durch den Einsatz von Näherungssensoren auf das Verhalten der Stakeholder reagieren kann. Hierdurch kann die Software beispielsweise proaktiv zwischen verschiedenen Darstellungs- und Eingabemodi wechseln und selbständig freien Platz auf der Zeichenfläche zum Schreiben neuer Inhalte zur Verfügung stellen. Die Autoren diskutieren hierbei insbesondere die Herausforderungen, auf unerwartetes Verhalten der Stakeholder reagieren und mit falsch interpretierten Absichten umgehen zu müssen. Darüber hinaus erläutern sie, wie ein solches System sinnvoll die erkannten Intentionen der Benutzer widerspiegeln, reaktive Prozessabläufe verständlich darstellen und Stakeholdern die Möglichkeit geben kann, proaktive Aktionen zu unterbrechen.

Chung et al. [45] präsentieren mit *InkKit* ein erweiterbares Toolkit zur Entwicklung von Sketching-Tools für mobile Endgeräte. *InkKit* stellt dabei sowohl grundlegende Basisfunktionalitäten wie Undo und Redo als auch komplexere Features wie Handschrift- und Mustererkennung zur Verfügung. Es kann einfache Formen wie Rechtecke und Kreise erkennen und diese zur Verschönerung durch computergenerierte Formen ersetzen. Zur Visualisierung von Beziehungen erlaubt die Anwendung darüber hinaus die manuelle Verknüpfung von logisch zusammenhängenden Skizzen in einem speziellen Storyboard-Modus.

Klemmer et al. [129] stellen mit *Outpost* einen Ansatz vor, der elektronische Whiteboards mit physischen Medien kombiniert. Die Autoren verwenden hierzu große Multi-Touch-Displays, die sie mit mehreren Kameras erweitern. Auf diese Weise können Stakeholder auf dem elektronischen Whiteboard digitale Skizzen anfertigen und diese zusätzlich mit analogen Haftnotizen versehen. Die physischen Klebezettel werden dabei von den Kameras erfasst und digitalisiert. Des Weiteren können die analogen Notizen manuell durch digital gezeichnete Linien auf dem elektronischen Whiteboard miteinander verbunden werden, um informelle Beziehungen zwischen diesen auszudrücken.

Mynatt et al. [176, 175] präsentieren mit *Flatland* ein Werkzeug für elektronische Whiteboards, das Anwender insbesondere bei informeller Büroarbeit unterstützen soll. *Flatland* ermöglicht die Erstellung digitaler Freihandskizzen und stellt Stakeholdern dabei eine Zeitleiste zur Verfügung, über die sie schnell zu relevanten älteren Ständen einer Skizze zurückkehren können. Hierzu sichert das Tool die skizzierten Inhalte kontinuierlich zu potentiell interessanten Zeitpunkten, beispielsweise vor einem längeren Zeitraum ohne Benutzereingaben oder vor der Entfernung größerer Segmente einer Skizze. Darüber hinaus stellt *Flatland* sogenannte „Behaviors“ zur Verfügung. Hiermit können bestimmte Teile einer Skizze markiert werden, um ihnen eine besondere Semantik zu verleihen. Entsprechend markierte Linienzüge lassen sich somit beispielsweise als vertikale ToDo-Listen anordnen oder für Kalkulationen als Spalten von Zahlen interpretieren, die automatisch addiert oder subtrahiert werden können.

Damm et al. [50] haben Softwareingenieure bei ihrer Arbeit beobachtet und daraus wichtige Kriterien für das Design objektorientierter Modellierungswerkzeuge abgeleitet. Basierend auf ihren Erkenntnissen haben sie mit *Knight* ein Werkzeug entwickelt, das ähnlich wie *Tahuti* zwei verschiedene Eingabemodi zur Verfügung stellt: Im Freihandmodus können Stakeholder beliebige Freihandskizzen erstellen. Diese werden exakt so dargestellt, wie sie gemalt wurden. Im UML-Modus werden alle gezeichneten Linien und Formen unmittelbar durch passende UML-Elemente ersetzt. Die erstellten Diagramme können jedoch trotzdem unvollständig bleiben und müssen keiner strikten UML-Semantik folgen.

Johnson et al. [114] geben eine verständliche Übersicht über weitere populäre Ansätze.

2.2.2. Erstellung von digitalen UI-Skizzen

Obrenovic und Martens [178] motivieren, dass die Skizzierung von statischen UIs alleine nicht ausreichend ist, um Stakeholdern ein adäquates Gefühl für das Look and Feel einer Anwendung zu vermitteln. Deshalb stellen sie mit *Sketchify* ein Werkzeug

zur Erstellung von User-Interfaces vor, das freihändiges Skizzieren mit traditionellen Programmierwerkzeugen für Endbenutzer wie Spreadsheets und Skripten kombiniert. Hard- und Softwarekomponenten, die Teil der interaktiven User Experience sind, wie beispielsweise Sensoreingaben sowie Audio- und Bildmaterial, werden von den Autoren als „interaktives Material“ bezeichnet. Anwender können sogenannte „interaktive Skizzen“ erstellen, indem sie interaktives Material manipulieren und mit Elementen von Freihandzeichnungen kombinieren.

Memmel and Reiterer [164] präsentieren mit *Inspector* ein Werkzeug zur Spezifikation von UIs für interaktive Systeme, das die Erstellung von informellen UI-Skizzen auf digitalen Whiteboards ermöglicht. Stakeholder können die erstellten Skizzen mit verschiedenen geometrischen Formen, Bildern und Widgets anreichern. Darüber hinaus erlaubt *Inspector* die manuelle Definition von Verknüpfungen zwischen grafischen Benutzerschnittstellen, textuellen Anforderungen und Geschäftsprozessen über verschiedene Abstraktionsebenen hinweg.

Ein komplexerer Ansatz wird von de Souza Alcantara et al. [52] vorgeschlagen: Sie präsentieren ein Werkzeug namens *ProtoActive* zur Erstellung von Low-Fidelity-Prototypen für Multi-Touch-basierte Anwendungen. *ProtoActive* enthält dabei eine Machine-Learning-Komponente namens *Intelligent Gesture Toolkit* (kurz: *IGT*). Diese ermöglicht die Definition individueller Gesten durch Training des Systems mit Beispielen. Hierdurch können insbesondere auch Anwender ohne Programmierkenntnisse leicht anwendungsspezifische Gesten definieren. Hosseini-Khayat et al. [107] haben *IGT* in ein System namens *Active Story Touch* (kurz: *AST*) integriert. Stakeholder können mit *AST* grafische Benutzerschnittstellen skizzieren und diese über Gesten, die zuvor in *IGT* definiert wurden, manuell miteinander verknüpfen. Die Gesten können danach in einem Simulationsmodus getestet werden und steuern die Dialogübergänge in dem so skizzierten Prototyp.

Coyette et al. [49] präsentieren ein Freihandzeichenwerkzeug namens *SketchiXML* für schnelles Interface-Design und Prototyping. Es ermöglicht die Erstellung informeller UI-Skizzen, die zur Weiterverarbeitung in plattformunabhängige XML-Dateien exportiert werden können. Hierzu verwendet *SketchiXML* eine Mustererkennung, die alle gezeichneten Formen analysiert und interpretiert. Es kann dabei die handge-

zeichneten Elemente entweder unverändert darstellen, wie sie von den Anwendern gezeichnet wurden, oder alle erkannten Elemente durch benutzerdefinierte Formen ersetzen.

Landay und Myers [136, 138] präsentieren mit *SILK* ein Werkzeug zur Erstellung von UI-Skizzen, das die Vorteile von Papierzeichnungen mit denen elektronischer Werkzeuge kombinieren soll. Stakeholder können Freihandzeichnungen von grafischen Benutzerschnittstellen anfertigen und manuell auf einem Storyboard zueinander in Beziehung setzen. Obwohl alle Skizzen als Freihandzeichnungen angefertigt werden, kann *SILK* verschiedene rudimentäre UI-Elemente wie Buttons, Checkboxen und Textfelder identifizieren. Hierdurch können Stakeholder das Verhalten handgezeichneter UI-Elemente spezifizieren und beispielsweise einen Button definieren, der einen Übergang von einer UI-Skizze zu einer anderen auslöst. Das Storyboard visualisiert die Zusammenhänge zwischen den Skizzen und zeigt an, wie Stakeholder zwischen ihnen navigieren können. Die so definierten Interaktionen können in einem speziellen Ausführungsmodus getestet werden, um den Stakeholdern ein besseres Gefühl für das Look and Feel der Applikation zu vermitteln. Darüber hinaus erlaubt *SILK* die Generierung von Quellcode für verschiedene Programmiersprachen zur Erzeugung der skizzierten UIs.

Ein ähnliches Werkzeug wird von Landays Studenten Newman and Lin [146, 177] vorgestellt: Sie präsentieren ein Tool namens *DENIM* zur Erstellung von interaktiven Skizzen für Webseiten. Diese können per Stifteingabe auf mobilen Geräten gezeichnet werden. *DENIM* bietet eine „Page“-Ansicht zur Erstellung individueller Skizzen sowie eine „Storyboard“-Ansicht zur manuellen Definition von Verknüpfungen zwischen diesen an. Skizzen können dabei über zwei Arten von Verknüpfungen miteinander verbunden werden: Ein „Navigational Link“ repräsentiert eine Transition zwischen zwei Skizzen, die durch ein skizziertes UI-Element wie beispielsweise einen Hyperlink ausgelöst werden kann. Ein „Organizational Link“ stellt eine konzeptuelle Beziehung zwischen zwei Webseiten dar. Ähnlich wie *SILK* erlaubt auch *DENIM* einen Test der definierten Dialogübergänge in einem speziellen Ausführungsmodus. Hierzu wird ein vereinfachtes Browser-Fenster gestartet, das die erstellten UI-Skizzen anzeigt und es Stakeholdern ermöglicht, zwischen diesen über die zuvor definierten Navigational Links zu navigieren.

Plimmer und Apperley [188, 189] präsentieren mit *Freeform* ein weiteres Werkzeug zur Skizzierung von grafischen Benutzerschnittstellen. *Freeform* nutzt Mustererkennung zur Analyse und Interpretation der gezeichneten Skizzen, um daraus Quellcode für die Programmiersprache Visual Basic zu generieren. In einer Studie zeigen die Autoren, dass digitale Tools zur Erstellung informeller Skizzen grundsätzlich zahlreiche Vorteile gegenüber klassischen Widget-basierten Designwerkzeugen sowie der traditionellen Skizzierung mit Stift und Papier haben.

Bailey und Konstan [14] präsentieren *DEMAIS* zur Erstellung von UI-Skizzen auf mobilen Endgeräten. Die erstellten Skizzen können mit dynamischen Medien in Form von Audio, Video und Animationen angereichert und zur Spezifikation des Dialogflusses manuell auf einem Storyboard miteinander verknüpft werden. *DEMAIS* kann die angefertigten Skizzen in einen ausführbaren Prototypen transformieren, um den Stakeholdern einen Eindruck des skizzierten interaktiven und zeitlichen Verhaltens zu vermitteln. In einer Studie konnten die Autoren zeigen, dass Anwender informelle Notationen, wie sie in *DEMAIS* zur Anwendung kommen, grundsätzlich strikten formalen Sprachen vorziehen.

Caetano et al. [37] stellen mit *JavaSketchIt* ein Java-Tool zur Erstellung von simplen Benutzeroberflächen vor. Stakeholder können UIs mit Freihandformen skizzieren, die durch Anwendung von Gesten- und Mustererkennung in passende vordefinierte Formen umgewandelt werden. Die gezeichneten Skizzen können danach in Java-Code transformiert werden, um auf diese Weise ausführbare Prototypen zu erzeugen. *JavaSketchIt* ermöglicht dabei jedoch keine Spezifikation von UI-Verhalten und Dialogflüssen. Die Autoren vergleichen in einer abschließenden Studie ihr Werkzeug mit einem kommerziellen UI-Builder und kommen zu dem Schluss, dass Stakeholder durch informelles freihändiges Skizzieren von UI-Skizzen schneller verwertbare Ergebnisse erzielen und diesen Prozess dabei als komfortabler und intuitiver erachten als die Bedienung einer entsprechenden Desktop-Anwendung mit Tastatur und Maus.

2.2.3. Verwaltung von digitalen Skizzen

Wie in Abschnitt 1.3 erläutert, ist die effiziente Verwaltung von Artefakten in komplexen Softwareprojekten eine wesentliche Herausforderung. Aus diesem Grund befassen sich einige Ansätze explizit mit der Verwaltung von digitalen Skizzen und deren sinnvoller Darstellung.

Mangano et al. [156] schlagen hierfür eine hierarchische Ansicht vor: Sie verwenden ein zweidimensionales Gitter, um Skizzen in logischen Gruppen, sogenannten „Clustern“, zu organisieren. Skizzen innerhalb des selben Clusters werden automatisch in einer konzentrischen Anordnung dargestellt und Stakeholder können diese manuell miteinander verknüpfen, um inhaltliche und logische Zusammenhänge darzustellen. Diese Beziehungen werden durch beschriftete Pfeile zwischen den Skizzen visualisiert.

Guimbretière et al. [90] schlagen zur Verwaltung von erstellten Skizzen einen ortsbasierten Skalierungsmechanismus namens „ZoomScapes“ vor. Die Grundidee besteht darin, alle Inhalte in sogenannten „Sheets“ zu platzieren. Hierbei handelt es sich um transparente rechteckige Bereiche, die frei auf dem Whiteboard umher bewegt und modifiziert werden können. Jedes Sheet kann dabei beliebige Artefakte wie Skizzen, Bilder und sogar aktive Anwendungen enthalten. Die Skalierung eines Sheets wird in Abhängigkeit von seiner aktuellen Position auf dem Whiteboard bestimmt und beim Verschieben dynamisch angepasst.

Ein ähnlicher Ansatz wird von Mynatt et al. [175] verfolgt: Ihr Tool *Flatland* fasst skizzierte Inhalte automatisch zu sogenannten „Segmenten“ zusammen, die frei auf dem Whiteboard verschoben werden können. Die Größe dieser Segmente wird automatisch reduziert, wenn sie am Rand des Whiteboards platziert werden. *Flatland* verkleinert hierbei insbesondere solche Segmente, mit denen die Stakeholder am längsten nicht mehr gearbeitet haben.

Lin et al. [145] verwenden zur Verwaltung der in ihrem Tool *DENIM* erstellten Freihandskizzen eine hierarchische Darstellung der Inhalte auf drei Ebenen. Die oberste Ebene ist die „Site Map“, die eine Übersicht über alle Skizzen präsentiert, indem diese als beschriftete Miniaturbilder angezeigt werden. Stakeholder können

die Vorschaubilder auf der Site Map frei verschieben und beliebig anordnen. Die mittlere Ebene ist das „Storyboard“, welches eine Auswahl mehrerer Skizzen und insbesondere die zwischen ihnen definierten Verknüpfungen zur Navigation zeigt. Die unterste Ebene ist das „Sketch Level“. Auf diesem werden die Skizzen in ihrer Originalgröße dargestellt und können von den Stakeholdern bearbeitet werden.

Streitz et al. [222] organisieren Skizzen und andere relevante Artefakte als Knoten innerhalb eines Graphen. Jeder Knoten kann hierbei eine beliebige Anzahl von Artefakten sowie auch andere Knoten enthalten und an einer beliebigen Stelle auf dem Whiteboard platziert werden. Knoten können manuell über Kanten miteinander verknüpft werden, um inhaltliche Beziehungen abzubilden. Die Inhalte können dabei wahlweise in einem privaten oder öffentlichen Arbeitsbereich abgelegt werden, auf den entweder nur ein spezifischer Nutzer oder alle Stakeholder eines Projekts Zugriff haben.

Stefik et al. [219, 220] schlagen zur Verwaltung von handgezeichneten Skizzen auf einem elektronischen Whiteboard vor, diese als Miniaturbilder in Form eines horizontalen Filmstreifens am unteren Rand des Bildschirms zu platzieren. Sie bezeichnen dieses Steuerelement als „Stampsheet“. Jedes Miniaturbild darin stellt in verkleinerter Form den Inhalt einer Freihandskizze dar, die auf der darüber positionierten Zeichenfläche erstellt wurde. Durch Selektion einer Position im Filmstreifen wird die entsprechende Skizze wiederhergestellt und auf der Zeichenfläche angezeigt, während die aktuell in Bearbeitung befindliche Skizze als Miniaturbild an der ausgewählten Stelle abgelegt wird.

2.3. Klassifizierung von Linienzügen

Damit die Inhalte der im AugIR befüllten Canvases interpretiert und nutzbar gemacht werden können, muss auf den gezeichneten Linienzügen eine Handschrifterkennung durchgeführt werden.

Die Ursprünge der Forschung im Bereich der Texterkennung (auch Optische Zeichenerkennung, engl. Optical Character Recognition, OCR) reichen zurück bis ins Jahr

1870, als der amerikanische Erfinder Charles R. Carey den ersten Retina-Scanner erfand [201]. Ein wichtiger Teilbereich der Texterkennung ist die computergestützte Handschrifterkennung, deren Ziel die möglichst fehlerfreie Erkennung von beliebigen handschriftlichen Texten ist [187]. Viele der heutzutage eingesetzten Algorithmen wurden bereits vor langer Zeit entwickelt [34, 147, 166].

Die Qualität der heutigen Handschrifterkenner ist hoch und es existieren zahlreiche kommerzielle sowie freie Bibliotheken, die sich problemlos an eigene Anwendungen anbinden lassen. Um jedoch auf den Linienzügen einer Freihandskizze eine Handschrifterkennung durchführen zu können, müssen die zu analysierenden Linienzüge zunächst identifiziert und aus der Menge aller gezeichneten Linienzüge ausgewählt werden, damit sie in den Handschrifterkenner eingegeben werden können. Man unterscheidet hierbei zwischen Text- und Form-Linienzügen [25, 182]: Ein *Text-Linienzug* ist ein Linienzug in einer Freihandskizze, der ein Zeichen oder einen Teil davon darstellt. Demgegenüber repräsentiert ein *Form-Linienzug* eine geometrische Form oder einen Teil davon. Weil nur Text-Linienzüge von einem Handschrifterkenner erkannt und sinnvoll interpretiert werden können, ist ein Vorverarbeitungsschritt notwendig, der in betrachteten Freihandskizzen Text-Linienzüge und Form-Linienzügen voneinander trennt. Dies ist jedoch ein nicht-triviales und grundsätzlich schwer zu lösendes Problem [238]. Die große Anzahl existierender Forschungsarbeiten zu diesem Thema und die verschiedenen Herangehensweisen legen nahe, dass bisher noch keine perfekte allgemeingültige Lösung existiert [54].

Im Folgenden werden ausgewählte populäre Ansätze zur Klassifizierung von Linienzügen betrachtet, auf deren Basis in Abschnitt 3.4 ein geeignetes Verfahren zur Trennung zwischen Text- und Form-Linienzügen in Interaction-Room-Canvases entwickelt wird.

Fast allen Arbeiten liegt hierbei die gemeinsame Idee zugrunde, dass Text- und Form-Linienzüge unterschiedliche charakteristische Eigenschaften aufweisen, anhand derer sie unterschieden werden können [25, 195]. Machii et al. [152] haben beispielsweise festgestellt, dass sich die Länge eines Linienzugs oft als sinnvolles Unterscheidungsmerkmal eignet, weil Form-Linienzüge häufig länger sind als Text-Linienzüge. Weitere nützliche Kriterien sind beispielsweise die Abstände zwischen zwei aufein-

anderfolgenden Linienzügen [182], deren Krümmungsverhalten [196] oder die Dauer der entsprechenden Zeichenoperation [237].

Basierend auf dieser Grundidee identifizieren Patel et al. [182] insgesamt 46 relevante Eigenschaften, die sinnvoll zur Unterscheidung von Text- und Linienzügen verwendet werden können. Im Rahmen einer Studie analysieren die Autoren 1 519 Linienzüge hinsichtlich dieser Eigenschaften und definieren als Ergebnis einen Entscheidungsbaum, der zur Klassifizierung herangezogen werden kann.

In einer nachfolgenden Arbeit [25] erweitern die Autoren die Anzahl der betrachteten Eigenschaften von 46 auf 114. Sie erläutern, dass bestimmte Merkmale von Linienzügen wichtiger sind als andere und die Auswahl passender Eigenschaften stark vom gegebenen Problemkontext abhängt. Darüber hinaus gelangen sie zu der Erkenntnis, dass falsch gewählte Eigenschaften nicht nur keinen positiven Beitrag zur Klassifizierung leisten, sondern sogar schädlich für den Klassifizierungsprozess sein und die Erkennungsrate signifikant senken können. Die Autoren haben deshalb eine Bibliothek entwickelt, die bei der Auswahl geeigneter Eigenschaften für spezifische Probleme helfen soll.

Qin [191] verwendet ebenfalls einen Entscheidungsbaum zur Klassifizierung von Linienzügen in Freihandskizzen. Er betrachtet hierzu ihre konvexe Hülle, ihr Krümmungsverhalten sowie die Anzahl ihrer Selbstüberschneidungen. Der Autor verwendet diese Eigenschaften jedoch nicht zur Unterscheidung von Text und Form, sondern klassifiziert Linienzüge stattdessen als bestimmte Formen, beispielsweise als gerade Linien, Kreise, Halbkreise und Ellipsen. Weil diese Art der Klassifizierung in der Regel leichter fällt, kommt der vorgestellte Ansatz mit einer geringeren Zahl betrachteter Eigenschaften aus.

Ein weiterer Klassifizierungsansatz wird von Shilman und Viola [210] vorgestellt. Sie präsentieren ein Framework zur Erkennung und Gruppierung von Formen und Symbolen in Freihandskizzen. Hierzu werden alle Linienzüge als Knoten in einen Graph eingefügt. Jeweils zwei Knoten werden in diesem Graphen über eine Kante miteinander verbunden, wenn die Distanz zwischen den entsprechenden Linienzügen unterhalb eines bestimmten Schwellwerts liegt. Der vorgestellte Algorithmus betrachtet alle zusammenhängenden Teilgraphen und testet gewisse Eigenschaften

der Linienzüge, wie beispielsweise ihr Krümmungsverhalten und die Lage ihrer Endpunkte, um diese als vorher definierte Symbole zu klassifizieren oder als ungültige Kombination abzulehnen.

Häufig kommen zur Klassifizierung von Linienzügen auch Methoden des maschinellen Lernens zum Einsatz. Viele Ansätze verwenden hierzu *rekurrente neuronale Netze* (*RNN*) [133]. Diese enthalten insbesondere Verknüpfungen zwischen Neuronen einer Schicht zu Neuronen derselben oder einer vorangegangen Schicht, wodurch sie sich besser zur Verarbeitung von Sequenzen eignen als Feedforward-Netze. Zahlreiche Ansätze verwenden *Long short-term memory Netze* (*LSTM-Netze*) [106], bei denen es sich um eine spezielle Art von RNNs handelt. Der Vorteil von LSTM-Netzen liegt darin, dass sie im Gegensatz zu traditionellen rekurrenten Netzen leichter trainiert werden und insbesondere zeitlich verzögerte Effekte besser berücksichtigen können, was insbesondere für Klassifizierungsaufgaben nützlich ist.

LSTM-Netze werden beispielsweise von Van Phan und Nakagawa [237, 238] zur Klassifizierung von Linienzügen in handgeschriebenen Dokumenten verwendet. Die Autoren trainieren das Netz hierzu sowohl mit lokalen Eigenschaften der Linienzüge wie Länge und Dauer der Zeichenoperation als auch mit globalen Eigenschaften wie der durchschnittlichen Länge räumlicher Nachbarn. Zur Evaluation ihres Ansatzes wenden sie diesen auf offen zugängliche Freihandskizzen aus den Datenbanken Kondate [172] und IAMonDo [112] an.

Auch Otte et al. [181] verwenden rekurrente neuronale Netze zur Klassifizierung von Linienzügen. Sie vergleichen dabei insbesondere RNNs und LSTM-Netze miteinander. Sie kommen zu dem Schluss, dass sich beide Verfahren sehr gut zur Klassifizierung eignen, wobei jedoch für die untersuchte Datenbasis RNNs bis zu 15 mal schneller trainiert werden konnten als vergleichbare LSTM-Netze. Die Autoren vermuten darüber hinaus, dass sich sowohl LSTM-Netze als auch RNNs ebenfalls sehr gut zur Klassifizierung von Gesten verwenden lassen.

Statt klassischer LSTM-Netze verwenden Indermühle et al. [111] *bidirektionale Long short-term memory Netze* (*BLSTM-Netze*) zur Klassifizierung von Linienzügen. Hierbei handelt es sich um eine Erweiterung gewöhnlicher LSTM-Netze, bei der Daten sowohl vorwärts als auch rückwärts in das Netz eingespeist werden können. Dies ist

für die Klassifizierung von Linienzügen von Vorteil, weil die Art eines Linienzugs häufig nicht nur von den vorausgegangenen Daten sondern auch von den nachfolgenden abhängt.

Darüber hinaus existieren noch weitere Ansätze, die andere Techniken des maschinellen Lernens verwenden, wie beispielsweise Conditional Random Fields (CRFs) [57, 59, 135], Support Vector Machines (SVMs) [262] oder einen k-Nearest-Neighbor-Algorithmus (kNN) [248]. Es gibt auch Arbeiten, die erfolgreich verschiedene Arten von Klassifizierungsansätzen miteinander kombinieren, um dadurch das erzielte Ergebnis zu verbessern [247]. Eine Übersicht über weitere aktuelle Arbeiten zur Klassifizierung von Linienzügen wird von Degtyarenko et al. [54] gegeben.

2.4. Traceability

Unter *Traceability* versteht man die Erfassung, Verwaltung und Nutzbarmachung von Zusammenhängen zwischen Artefakten in einem Softwareprojekt [46]. Die erfassten Zusammenhänge werden als *Trace Links* bezeichnet [46].

Durch eine Identifikation und Verknüpfung von zusammenhängenden Artefakten können Stakeholder bei der horizontalen Navigation zwischen diesen unterstützt werden. Im Allgemeinen unterscheidet man hierbei zwischen prospektiver und retrospektiver Traceability [11, 46]:

- *Prospektive Traceability-Techniken* beobachten die Aktivitäten der Stakeholder und erfassen Trace Links zwischen Artefakten in situ, während die Stakeholder mit diesen arbeiten. Dies ermöglicht insbesondere die Betrachtung zeitlicher Abläufe und kontextueller Zusammenhänge, die ansonsten mit anderen Traceability-Methoden nicht erfasst werden können [12]. Prospektive Techniken sind dabei für gewöhnlich manuell oder nur teilweise automatisiert [11].
- *Retrospektive Traceability-Techniken* analysieren statische Mengen existierender Artefakte und erfassen Trace Links zwischen diesen basierend auf den extrahierten Informationen. Die meisten automatisierten Traceability-Verfahren

fallen in diese Kategorie [11]. Viele davon verwenden Techniken des Information Retrievals, die auf der Auswertung textueller Inhalte basieren [77].

Für den AugIR sind insbesondere retrospektive Ansätze von Interesse, weil sich hierdurch automatisiert Verknüpfungen zwischen zusammenhängenden Canvases erfassen lassen. Durch die entsprechenden Trace Links können Stakeholder sinnvoll bei der horizontalen Navigation zwischen den erstellten Freihandskizzen unterstützt werden. Grundsätzlich nimmt der Bedarf für eine Automatisierung von Traceability-Techniken mit steigender Komplexität und Anzahl der Artefakte in einem Projekt zu [11].

Häufig kommt bei retrospektiven Ansätzen das von Deerwester et al. [53] entwickelte *Latent Semantic Indexing* (*LSI*) zur Erfassung von Trace Links zum Einsatz. Die mathematische Grundidee hierbei ist Folgende [66]: Es wird angenommen, dass in zueinander inhaltlich ähnlichen Texten die gleichen oder ähnliche Wörter vorkommen. Für ein betrachtetes Dokument wird deshalb eine sogenannte Term-Dokument-Matrix erzeugt, die genau eine Zeile für jedes im Dokument vorkommende Wort und eine Spalte für jeden Textabschnitt enthält. Jedes Element der Matrix gibt an, wie häufig das entsprechende Wort im jeweiligen Abschnitt des Dokuments erscheint. Durch Anwendung der sogenannten *Singular Value Decomposition* (*SVD*) wird die Dimension dieser Matrix zunächst soweit wie möglich reduziert, bevor die paarweise Ähnlichkeit zwischen jeweils zwei Wörtern – jedes davon repräsentiert durch eine Zeile der Matrix – bestimmt wird. LSI eignet sich besonders gut zur Analyse großer Textdokumente und lässt sich grundsätzlich auch zur Verknüpfung unterschiedlicher Dokumente verwenden, indem statt einzelner Textabschnitte jeweils ein komplettes Dokument für jede Spalte der Matrix betrachtet wird.

Für die Analyse kürzerer Texte, wie sie im Interaction-Room-Kontext vorkommen, eignet sich in der Regel das von Salton et al. [198] entwickelte *Vector Space Model* (*VSM*) besser, weil es im Vergleich zu LSI einfacher zu verstehen und anzuwenden ist sowie darüber hinaus signifikante Geschwindigkeitsvorteile hat. Die Grundidee ist jedoch die gleiche: Jedes Dokument wird durch einen numerischen Vektor beschrieben, der anzeigt, wie häufig ein bestimmtes Wort in diesem Dokument vorkommt.

Zwei Dokumente werden als ähnlich betrachtet, wenn sie die gleichen Wörter in ähnlicher Häufigkeit enthalten. Je ähnlicher also zwei Dokumente sind, desto ähnlicher sind folglich die charakterisierenden Vektoren. Sowohl LSI als auch VSM nutzen zur Bestimmung der Ähnlichkeit zwischen zwei Vektoren die Kosinus-Ähnlichkeit, die im Detail in Abschnitt 3.2.8 beschrieben wird.

Viele Traceability-Ansätze widmen sich der Verknüpfung von Quellcode mit textuellen Dokumenten. Marcus et al. [159, 160] nutzen beispielsweise LSI zur Erfassung von Trace Links zwischen Quellcode und Texten aus Anforderungs- und Designdokumenten. Hierzu extrahieren und analysieren sie Kommentare und Bezeichner aus dem Quellcode eines betrachteten Projekts und erstellen unter zusätzlicher Berücksichtigung der Textdokumente wie eingangs beschrieben eine Term-Dokument-Matrix. Die Autoren führen zwei Experimente durch, indem sie existierende Projekte analysieren und ihre Ergebnisse mit ähnlichen Ansätzen vergleichen, die andere Information-Retrieval-Techniken einsetzen. Sie kommen zu dem Schluss, dass die Verwendung von LSI in diesem Kontext sehr gute Ergebnisse erzielt.

Antoniol et al. [10] verwenden für einen ähnlichen Zweck VSM. Auch sie erfassen Trace Links zwischen Quellcode und textuellen Dokumenten. Sie führen hierzu zwei Fallstudien durch und vergleichen dabei VSM mit einem probabilistischen Traceability-Ansatz. Die Autoren zeigen, dass beide Ansätze ähnlich gute Ergebnisse liefern, wobei die Verwendung von VSM mit weniger Aufwand verbunden ist.

Chen et al. [41, 42] kombinieren Information-Retrieval-Methoden mit verschiedenen Techniken wie regulären Ausdrücken und Key Phrases um Geschwindigkeit und Genauigkeit der Trace-Link-Erfassung zu erhöhen. Auch sie stellen hierzu Verknüpfungen zwischen Quellcode und Textdokumenten her. Sie wenden ihren Ansatz auf den Sourcecode des Java Development Kits an und vergleichen die erzielten Ergebnisse mit VSM. Dabei kommen sie zu dem Schluss, dass ihr vorgestellter Ansatz bei Verwendung hoher Schwellwerte im Vergleich zu VSM bessere Ergebnisse liefert, während letzteres für niedrige Schwellwerte zu präferieren ist.

Darüber hinaus existieren auch zahlreiche Forschungsarbeiten, die sich mit der Erfassung von Trace Links zwischen Textdokumenten und formalen Modellen befassen. Spanoudakis [216] stellt beispielsweise einen Ansatz vor, der automatisch Trace

Links zwischen Anforderungsdokumenten und UML-Modellen erfassen kann. Er nutzt hierzu heuristische Regeln, die eine Zuordnung von Elementen in Modellen zu textuellen Abschnitten in Anforderungsdokumenten beschreiben.

Auch Letelier [141] widmet sich der Erfassung von Trace Links zwischen Anforderungen und formalen UML-Diagrammen. Er definiert hierzu ein passendes Metamodell zur Nachverfolgung von Anforderungen und stellt ein entsprechendes Framework vor, das eine einheitliche Repräsentation von Softwareentwicklungsartefakten und den Trace Links zwischen ihnen ermöglicht.

Settimi et al. [207] präsentieren einen interessanten Ansatz, der beides kombiniert: Anstatt Quellcode und Textdokumente direkt miteinander zu verknüpfen, werden Information-Retrieval-Methoden verwendet, um über UML-Diagramme indirekte Verbindungen zwischen diesen Artefakten herzustellen.

Zusätzlich zur initialen Erfassung von Trace Links muss auch deren Verwaltung berücksichtigt werden. Wenn sich Inhalte von Artefakten ändern, können dadurch neue Verknüpfungen zu anderen Artefakten entstehen oder bereits erfasste Trace Links können obsolet werden. Insbesondere können Änderungen an einem Artefakt auch Auswirkungen auf mit diesem verknüpfte Artefakte nach sich ziehen. Diese sogenannte *Impact Analysis* ist oft nicht trivial und kann für komplexe Softwareprojekte eine große Herausforderung darstellen [173, 200, 217].

Eine umfassende Übersicht über weitere interessante Forschungsarbeiten zum Thema Traceability wird von Winkler und Pilgrim [250] gegeben.

2.5. Bewertung der verwandten Arbeiten

In Abschnitt 2.1 wurde deutlich, dass Team-Räume grundsätzlich sinnvoll durch elektronische Komponenten erweitert werden können. Weil in fast allen analogen Team-Räumen traditionelle Whiteboards und Flip-Charts zum Einsatz kommen, überrascht es wenig, dass die meisten technisch augmentierten Räume mit elektronischen Whiteboards ausgestattet sind. Insbesondere neuere Ansätze ergänzen diese großen interaktiven Displays häufig durch den Einsatz mobiler Endgeräte, wodurch

sich gänzlich neue Möglichkeiten zur Unterstützung von Mitgliedern eines Softwareentwicklungsprojekts ergeben. Auch im AugIR sollen Tablets zur Erstellung digitaler Story Cards verwendet werden, um auf diese Weise Medienbrüche zu vermeiden und eine sinnvolle Digitalisierung des Feature Canvases zu ermöglichen.

In Abschnitt 2.2 wurden zahlreiche existierende Werkzeuge zur Erstellung digitaler Freihandskizzen präsentiert. Jedoch stellen nur wenige Ansätze mehrere Perspektiven mit unterschiedlichen Semantiken zur Verfügung. Darüber hinaus unterstützen die meisten Werkzeuge nur ein einzelnes großes Display, während im AugIR insbesondere mehrere miteinander verbundene elektronische Whiteboards zum Einsatz kommen sollen, die es ermöglichen, ein Softwareprojekt gleichzeitig aus verschiedenen Blickwinkeln zu betrachten. Dies ist insbesondere für Stakeholder wichtig, die an Benutzerschnittstellen oder komplexen Softwarearchitekturen arbeiten [89, 185].

Einige vorgestellte Werkzeuge ersetzen die handgezeichneten Elemente in den erstellten Skizzen sofort durch computergenerierte Formen und Symbole und erzeugen hierdurch bereits zur Laufzeit semiformale Diagramme. Dieses Vorgehen erscheint jedoch für den Interaction-Room-Kontext nicht sinnvoll, weil sich dies gemäß der in Abschnitt 1.1 diskutierten Überlegungen negativ auf den gemeinschaftlichen kreativen Austausch der Stakeholder auswirken kann.

Viele Werkzeuge zur Erstellung von UI-Skizzen stellen einen Simulationsmodus zur Verfügung, weil die Skizzierung von statischen UIs alleine nicht ausreichend ist, um Stakeholdern ein angemessenes Gefühl für das Look and Feel einer Anwendung zu vermitteln [178]. Sie ermöglichen dadurch eine interaktive Ausführung der gezeichneten Benutzeroberflächen und eine Veranschaulichung der Dialogflüsse. Eine solche Interaktivität erscheint auch für den Interaction Canvas nützlich und soll deshalb im AugIR ebenfalls zum Einsatz kommen.

Die meisten der in Abschnitt 2.2 vorgestellten Werkzeuge unterstützen zur Definition der Interaktivität jedoch entweder gar keine Gesten oder beschränken sich auf rudimentäre Standardgesten. Aufgrund der steigenden Verbreitung von mobilen Endgeräten und Multi-Touch-Displays wird aber die Betrachtung von komplexen Gesten zunehmend wichtiger [102]. Der digitale Interaction Canvas im AugIR soll

deshalb insbesondere auch nicht-technischen Stakeholdern die einfache und intuitive Definition benutzerdefinierter Gesten zur Spezifikation von Anwendungsverhalten und Dialogübergängen ermöglichen.

In Abschnitt 1.3 wurde bereits erläutert, warum eine Unterstützung der Stakeholder bei der Navigation zwischen Artefakten sinnvoll erscheint. Aus diesem Grund existieren zahlreiche Traceability-Ansätze, die sich mit der (automatischen) Erfassung von Zusammenhängen zwischen Artefakten befassen. Auch viele Sketching-Tools unterstützen deshalb die Verknüpfung von Skizzen. Es ist jedoch auffällig, dass diese Beziehungen ausschließlich manuell spezifiziert werden können. Es existieren zwar zahlreiche Ansätze zur automatischen Erfassung von Trace Links, diese beschränken sich jedoch in der Regel auf die Erfassung von Zusammenhängen zwischen Quellcode und textuellen Dokumenten oder zwischen Texten und formalen Diagrammen. Es ist kein Ansatz bekannt, der automatisch entsprechende Verknüpfungen zwischen informellen Freihandskizzen erfasst. Ein solcher Automatismus kann Stakeholder jedoch bei der Identifikation inhaltlicher Zusammenhänge, der Navigation zwischen Artefakten sowie der Aufdeckung von Widersprüchen und potentiellen Projektrisiken unterstützen. Im AugIR soll deshalb eine retrospektive Traceability-Technik realisiert werden, die Trace Links zwischen Freihandskizzen basierend auf deren textuellen Inhalten erfasst. Hierzu soll VSM verwendet und durch eine unscharfe Suche erweitert werden, um möglichen Fehlern bei der Handschrifterkennung entgegenzuwirken.

Weiterhin kann festgestellt werden, dass sich fast alle in Abschnitt 2.2.2 vorgestellten Arbeiten zur expliziten Skizzierung von Benutzeroberflächen auf UI-Skizzen beschränken und die Zusammenhänge zwischen skizzierten Dialogen und anderen Diagrammtypen wie beispielsweise Geschäftsprozessskizzen vollständig ignorieren. Eine Betrachtung dieser Beziehungen kann jedoch abhängig vom Projektkontext durchaus relevant sein, weshalb der AugIR durch Einsatz von Traceability insbesondere auch Skizzen mit unterschiedlicher Semantik miteinander verknüpfen soll [127].

Darüber hinaus wurde in Abschnitt 2.3 die Notwendigkeit einer Klassifizierung von Linienzügen in Freihandskizzen zur Realisierung einer Handschrifterkennung moti-

viert. Alle vorgestellten Ansätze verfolgen hierbei die gleiche Strategie, indem sie Linienzüge anhand bestimmter repräsentativer Eigenschaften klassifizieren. Hierzu kommen verschiedene Techniken wie Entscheidungsbäume und Methoden des maschinellen Lernens zum Einsatz, die jeweils unterschiedliche Vor- und Nachteile haben. Entscheidungsbäume sind grundsätzlich sehr leicht zu implementieren und anzuwenden. Jedoch haben beispielsweise Blagojevic et al. [25] gezeigt, dass eine große Anzahl potentiell relevanter Eigenschaften von Linienzügen betrachtet werden muss und eine Auswahl geeigneter Merkmale und passender Schwellwerte zur Definition eines solchen Entscheidungsbaums schwierig ist. Demgegenüber können Systeme durch den Einsatz von Methoden des maschinellen Lernens gut auf realen Daten trainiert werden und erzielen auch bei der Betrachtung einer geringeren Menge von Eigenschaften bei der Klassifizierung von Linienzügen häufig exzellente Ergebnisse. Viele Machine-Learning-Verfahren sind jedoch mathematisch hochkomplex und dadurch schwer zu verstehen und anzuwenden. Darüber hinaus bieten weder Entscheidungsbäume noch die bisher vorgestellten Ansätze zum maschinellen Lernen eine explizite Unterstützung zur Auswahl geeigneter Klassifizierungsmerkmale oder zur Festlegung passender Schwellwerte. Für den AugIR soll deshalb ein Verfahren entwickelt werden, dass nicht nur sehr gute Ergebnisse bei der Klassifizierung von Linienzügen erzielt, sondern auch sinnvoll bei der Auswahl geeigneter Eigenschaften unterstützt.

Zusammenfassend lässt sich somit festhalten, dass keiner der bisher vorgestellten Ansätze die in Abschnitt 1.3 diskutierten Probleme zufriedenstellend löst. Mit Hinblick darauf wird im nächsten Kapitel ein Konzept für einen Augmentierten Interaction Room präsentiert, der die genannten Einschränkungen und Nachteile beseitigt.

Teil II.

Lösung

3. Konzept des Augmentierten Interaction Rooms

Dieses Kapitel beschreibt das Konzept des Augmentierten Interaction Rooms. Zunächst werden die Anforderungen an den AugIR sowie ein entsprechendes Lösungskonzept kurz skizziert. Daraufhin werden wichtige Begriffe und Funktionen definiert, die zum Verständnis der weiteren Abschnitte dieses Kapitels notwendig sind. Danach werden verschiedene Algorithmen vorgestellt, mit denen die handgezeichneten Linienzüge in mehreren aufeinanderfolgenden Schritten verarbeitet und nutzbar gemacht werden können. Abschließend wird diskutiert, wie Freihandskizzen im AugIR erstellt und verwaltet, Stakeholder bei der Navigation zwischen skizzierten Inhalten unterstützt und potentielle Inkonsistenzen, Widersprüche und Projektrisiken durch Anwendung einer geeigneten Impact Analysis vermieden werden können.

3.1. Überblick über den Lösungsansatz

Aus den in Abschnitt 1.3 diskutierten Problemen, die zusammenfassend in Tabelle 1.2 aufgeführt sind, ergeben sich unmittelbar eine Reihe von Anforderungen, die in dem entwickelten Konzept des Augmentierten Interaction Rooms berücksichtigt wurden:

Der AugIR bietet eine umfangreiche Unterstützung für die Erstellung und Manipulation der handgezeichneten Inhalte auf den elektronischen Whiteboards. Dies umfasst beispielsweise die Möglichkeit, gezeichnete Linienzüge frei auf der Zeichenfläche zu verschieben und alle Änderungen beliebig rückgängig zu machen. Dabei steht den

Stakeholdern eine theoretisch unbegrenzte Anzahl von Zeichenflächen zur Verfügung, zwischen denen sie schnell und intuitiv wechseln können. Die Zeichenflächen sind in ihrer Größe unbeschränkt und lassen sich frei in alle Richtungen verschieben und stufenlos zoomen. Der Entwurf alternativer Lösungen wird unterstützt, indem alle skizzierten Inhalte auf Knopfdruck vervielfältigt werden können. Verschiedene Skizzen können dabei miteinander verglichen werden, indem sie auf dem selben elektronischen Whiteboard nebeneinander angeordnet oder simultan auf unterschiedlichen Whiteboards dargestellt werden. Eine zusätzliche Persistierung der Skizzen ist nicht notwendig, weil diese bereits in digitaler Form auf den elektronischen Whiteboards angefertigt werden. Das Konzept zur Erstellung und Bearbeitung von Skizzen wird im Detail in Abschnitt 3.9 erläutert.

Um hierbei Medienbrüche zu vermeiden, müssen alle Inhalte in digitaler Form entstehen. Während Skizzen bereits auf den digitalen Whiteboards gezeichnet werden können, werden zur Erfassung der Features für den Feature Canvas mobile Endgeräte mit Stifteingabe angebunden. Auf diese Weise können Stakeholder digitale Karteikarten auf einem Tablet schreiben und drahtlos an die elektronischen Whiteboards übertragen. Dies wird in Abschnitt 3.9.1 beschrieben.

Um darüber hinaus eine adäquate Verhaltensbeschreibung von Benutzerschnittstellen zu ermöglichen, wird der Interaction Canvas durch ein digitales Storyboard realisiert, welches die Verknüpfung von UI-Skizzen über benutzerdefinierte Gesten und deren interaktive Ausführung in einem Simulationsmodus ermöglicht. Dies wird in Abschnitt 3.9.2 thematisiert.

Zur Verwaltung der skizzierten Inhalte wird eine Graph-Datenstruktur verwendet, die in Abschnitt 3.10 präsentiert wird. Diese stellt alle Skizzen eines Projekts hierarchisch dar und ermöglicht die Verfeinerung von Skizzen sowie deren Zusammenfassung zu logischen Gruppen. Sie bietet überdies eine Unterstützung für die vertikale und horizontale Navigation zwischen Skizzen, wie es in Abschnitt 3.11 beschrieben wird.

Durch Anwendung einer geeigneten Impact Analysis kann darüber hinaus die inhaltliche Konsistenz platzierter Annotationen sichergestellt werden. Hierzu werden diese automatisch von einem annotierten Element auf alle anderen Elemente, die mit

diesem verknüpften sind, übertragen. Diese Technik ermöglicht zudem eine automatisierte Identifikation möglicher Widersprüche, die sich durch die Anwendung gegensätzlicher Annotationen ergeben. Dies wird in Abschnitt 3.12.1 erläutert. Weiterhin erlaubt die transitive Übertragung von Annotationen auf verbundene Elemente eine automatische Identifikation möglicher Projektrisiken, wie es in Abschnitt 3.12.2 erläutert wird.

Um die zuvor beschriebenen Lösungselemente realisieren zu können, ist eine umfangreiche Analyse und Interpretation der handgezeichneten Inhalte notwendig. Hierzu müssen diese in mehreren Schritten verarbeitet und insbesondere einer Handschrifterkennung unterzogen werden. Der gesamte Prozess ist in Abbildung 3.1 illustriert:

Die auf den Canvases erstellten Freihandskizzen enthalten sowohl Texte als auch Formen (wie beispielsweise Pfeile, Rauten und Rechtecke). Um die Inhalte dieser Skizzen interpretieren und sinnvoll nutzen zu können, ist deshalb zunächst eine Unterscheidung der gezeichneten Linienzüge notwendig. Aus diesem Grund erfolgt im ersten Schritt eine Klassifizierung jedes Linienzugs als Text- oder Form-Linienzug. Dies wird in Abschnitt 3.4 im Detail erläutert.

Im zweiten Schritt müssen die zuvor identifizierten potentiellen Text-Linienzüge zu Zeichen, Wörtern und gegebenenfalls vollständigen Sätzen zusammengefasst werden, um auf ihnen anschließend eine Handschrifterkennung durchführen zu können. Dieser Prozess wird in Abschnitt 3.5 beschrieben.

Um die erkannten Texte weiterverarbeiten und analysieren zu können, müssen diese im dritten Schritt in Tupel aufgetrennt werden, die jeweils ein oder mehrere Elemente umfassen. Hierbei kommen unterschiedliche heuristische Regeln zum Einsatz, die im Detail in Abschnitt 3.6 diskutiert werden. In der digitalen Sprachverarbeitung nennt man diesen Prozess *Tokenisierung* [233].

Im vierten Schritt werden die zuvor erzeugten Tokenisierungen analysiert und gegebenenfalls modifiziert, indem darin enthaltene Elemente durch hinreichend ähnliche

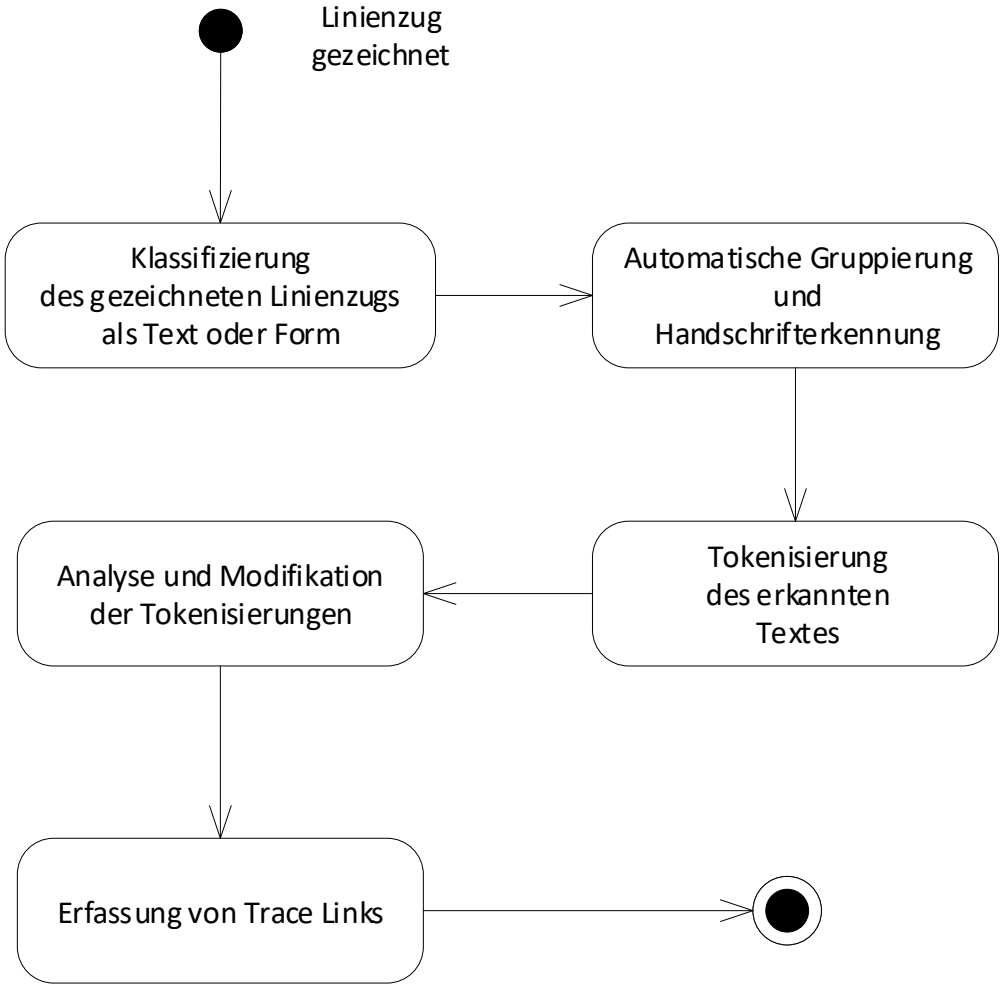

Abbildung 3.1.: Notwendige Schritte zur Analyse und Verarbeitung von Linienzügen.

Elemente ersetzt werden. Hierdurch kann die Korrelation potentiell zusammenhängender Elemente und damit die Erfassung textueller Beziehungen signifikant verbessert werden. Dies wird in Abschnitt 3.7 erläutert.

Im fünften und letzten Schritt werden textuell ähnliche Elemente verschiedener Skizzen durch Anwendung des Vector Space Models miteinander verknüpft. Dies ermöglicht beispielsweise eine direkte horizontale Navigation zwischen verbundenen Elementen und die Übertragung von Annotationen zur Wahrung der Konsistenz. Der entsprechende Traceability-Algorithmus wird in Abschnitt 3.8 präsentiert.

3.2. Definitionen und mathematische Grundlagen

In diesem Abschnitt werden zunächst einige zentrale Begriffe definiert, die für das weitere Verständnis notwendig sind. Darauf aufbauend werden wichtige Funktionen eingeführt, die an verschiedenen Stellen zur Anwendung kommen.

3.2.1. Definitionen zu Zeichenketten

Die Menge $\mathcal{A}$ sei definiert als die Menge aller Zeichen, die im Unicode-Zeichensatz [232] enthalten sind. Sie wird im Folgenden als *Alphabet* bezeichnet.

Die Menge $\mathcal{Z} \subset \mathcal{A}$ aller *Standardzeichen* sei definiert als

$$\mathcal{Z} := \{\text{„a“}, \ldots, \text{„z“}, \text{„A“}, \ldots, \text{„Z“}, \text{„0“}, \ldots, \text{„9“}, \text{„ä“}, \text{„ö“}, \text{„ü“}, \text{„Ä“}, \text{„Ö“}, \text{„Ü“}, \text{„ß“}\}.$$

Die Menge $\mathcal{Z}$ umfasst damit alle Buchstaben „a“ bis „z“ in Groß- und Kleinschreibweise, alle Ziffern „0“ bis „9“, die Umlaute „ä“, „ö“ und „ü“ in Groß- und Kleinschreibweise sowie das deutsche Eszett.

Die Menge $\mathcal{S} \subset \mathcal{A}$ aller *Sonderzeichen* sei definiert durch $\mathcal{S} := \mathcal{A} \setminus \mathcal{Z}$.

Das *Leerzeichen* spielt an vielen Stellen eine besondere Rolle und wird deshalb gesondert mit dem Symbol β ausgezeichnet. Es ist $\beta \in \mathcal{S}$.

Eine endliche Folge s beliebiger Zeichen aus $\mathcal{A}$ wird im Folgenden *Zeichenkette* genannt. Sie wird geschrieben als $s := (s_1, s_2, \ldots, s_n)$ oder $s :=$ „$s_1 s_2 \ldots s_n$“ mit $s_i \in \mathcal{A}$ für $i = 1, \ldots, n$.

Die *Länge* einer Zeichenkette s wird notiert als $|s|$. Sie gibt die Anzahl der Zeichen in dieser Zeichenkette an.

Die *Menge aller Zeichenketten der Länge n* über dem Alphabet $\mathcal{A}$ wird mit $\mathcal{A}^n$ bezeichnet. Die *Menge aller Zeichenketten beliebiger Länge* über dem Alphabet $\mathcal{A}$ wird mit $\mathcal{A}^*$ bezeichnet.

Mit dem Operator „+“ können zwei Zeichenketten miteinander verknüpft werden. Für $s := (s_1, \ldots, s_n)$ mit $|s| = n$ und $t := (t_1, \ldots, t_m)$ mit $|t| = m$ ist die *Verknüpfung* von s und t definiert durch $s+t := (s_1, \ldots, s_n, t_1, \ldots, t_m)$. Entsprechend gilt $|s+t| = |s| + |t| = n + m$.

Die *leere Zeichenkette* hat die Länge 0 und wird mit dem Symbol ε bezeichnet. Für Zeichenketten $s, t \in \mathcal{A}^*$ gilt $s + \varepsilon = \varepsilon + s = s$ sowie $s + \varepsilon + t = s + t$. Das An- oder Einfügen der leeren Zeichenkette hat dementsprechend keine Auswirkungen und verändert die entsprechende Zeichenkette nicht.

Die *Menge aller Tupel mit m Zeichenketten beliebiger Länge* über dem Alphabet $\mathcal{A}$ wird mit $\mathcal{A}^{*^m}$ bezeichnet. Für $u \in \mathcal{A}^{*^m}$ mit $u := (u_1, \ldots, u_m)$ ist also $u_j \in \mathcal{A}^*$ für $j = 1, \ldots, m$.

Um anzuzeigen, dass ein Zeichen $c \in \mathcal{A}$ in einer Zeichenkette $s := (s_1, \ldots, s_n) \in \mathcal{A}^*$ vorkommt, d. h. dass $\exists\ i \in \{1, \ldots, n\}$ mit $s_i = c$, wird im Folgenden $c \in s$ geschrieben. Entsprechend wird mit $c \notin s$ ausgedrückt, dass das Zeichen c nicht in der Zeichenkette s vorkommt.

Um anzuzeigen, dass eine Zeichenkette $s \in \mathcal{A}^*$ in einem Tupel $u := (u_1, \ldots, u_m) \in \mathcal{A}^{*^m}$ von Zeichenketten vorkommt, d. h. dass $\exists\ i \in \{1, \ldots, m\}$ mit $u_i = s$, wird im Folgenden $s \in u$ geschrieben. Entsprechend wird mit $s \notin u$ ausgedrückt, dass die Zeichenkette s nicht als Element im Tupel u vorkommt.

Eine Zeichenkette $w \in \mathcal{A}^*$ wird *Wort* genannt, wenn sie keine Leerzeichen enthält, wenn also gilt $\beta \notin w$.

3.2.2. Positionsbestimmung von Zeichen in Zeichenketten

Eine grundlegende Operation ist die Bestimmung aller Positionen eines gegebenen Zeichens innerhalb einer Zeichenkette. Um diese Operation zu definieren, wird zunächst eine Hilfsmenge $P(s, c)$ gebildet. Seien hierzu $s := (s_1, \ldots, s_n) \in \mathcal{A}^*$ eine Zeichenkette und $c \in \mathcal{A}$ ein Zeichen. Dann ist $P(s, c)$ gegeben durch

$$P(s, c) := \{i \in \mathbb{N} \mid s_i = c\}.$$

Die Menge $P(s, c)$ ist also die (ungeordnete) Menge aller Indizes, die die Positionen des Zeichens c in der Zeichenkette s angeben. Es ist $P(s, c) = \emptyset$, falls $c \notin s$.

Falls $c \in s$, lässt sich eine Funktion Pos aufbauend auf der zuvor eingeführten Menge $P(s, c)$ wie folgt definieren: Sei $m := |P(s, c)|$ die Anzahl der Elemente in $P(s, c)$, also die Häufigkeit des Vorkommens des Zeichens c in der Zeichenkette s. Dann ist $\text{Pos} : \mathcal{A}^* \times \mathcal{A} \to \mathbb{N}^m$ definiert als

$$\text{Pos}(s, c) := (i_1, \ldots, i_m),$$

wobei $i_j \in P(s, c)$ für $j = 1, \ldots, m$ und $i_k < i_{k+1}$ für $1 \leq k < m$.

$\text{Pos}(s, c)$ ist also die *sortierte Folge der Positionen*, an denen das Zeichen c in der Zeichenkette s vorkommt.

Es ist zu beachten, dass Pos nicht definiert ist, falls $c \notin s$.

3.2.3. Auftrennung von Zeichenketten

Häufig muss eine Zeichenkette an einem bestimmten Trennzeichen aufgespalten und in mehrere kleinere Zeichenketten zerlegt werden. Hierzu seien $s := (s_1, \ldots, s_n) \in \mathcal{A}^*$ eine Zeichenkette und $c \in \mathcal{A}$ ein Zeichen. Weiterhin sei $m := |P(s, c)|$ die Häufigkeit des Vorkommens des Zeichens c in der Zeichenkette s. Damit lässt sich eine Methode zur Auftrennung der Zeichenkette s am Trennzeichen c als $\text{Split} : \mathcal{A}^* \times \mathcal{A} \to \mathcal{A}^{*^{m+1}}$ durch $\text{Split}(s, c) := (w_1, \ldots, w_{m+1})$ wie folgt definieren:

Für $m = 0$ sei $\text{Split}(s, c) := (s)$, also das Tupel, welches nur das Element s enthält.

Für $m > 0$ seien die sortierten Positionen aller Vorkommen von c in s gegeben durch $\text{Pos}(s, c) = (i_1, \ldots, i_m)$. Dann ist

$$
\begin{aligned}
w_1 &:= \begin{cases} (s_1, \ldots, s_{i_1-1}), & \text{falls } i_1 > 1 \\ \varepsilon, & \text{sonst} \end{cases} \\
w_2 &:= \begin{cases} (s_{i_1+1}, \ldots, s_{i_2-1}), & \text{falls } i_2 > (i_1 + 1) \\ \varepsilon, & \text{sonst} \end{cases} \\
&\ldots \\
w_j &:= \begin{cases} (s_{i_{j-1}+1}, \ldots, s_{i_j-1}), & \text{falls } i_j > (i_{j-1} + 1) \\ \varepsilon, & \text{sonst} \end{cases} \\
&\ldots \\
w_{m+1} &:= \begin{cases} (s_{i_m+1}, \ldots, s_n), & \text{falls } n \geq (i_m + 1) \\ \varepsilon, & \text{sonst.} \end{cases}
\end{aligned}
$$

Es ist zu beachten, dass das von Split erzeugte Tupel leere Zeichenketten enthalten kann. Dies ist immer dann der Fall, wenn c das erste oder letzte Zeichen in der Zeichenkette s ist oder wenn es mehrmals direkt aufeinander folgt. Häufig ist dieser Effekt jedoch unerwünscht. Aus diesem Grund wird im Folgenden zusätzlich eine weitere Funktion definiert, die als Resultat ein Tupel mit ausschließlich nichtleeren Zeichenketten zurückliefert.

Hierzu sei k die Anzahl der $w_i \in \text{Split}(s, c) = (w_1, \ldots, w_{m+1})$ mit $w_i \neq \varepsilon$. Dann lässt sich eine Funktion Split^+ als Abbildung $\text{Split}^+ : \mathcal{A}^* \times \mathcal{A} \to \mathcal{A}^{*^k}$ mit $\text{Split}^+(s, c) := (w'_1, \ldots, w'_k)$ wie folgt definieren: Das Tupel $(w'_1, \ldots, w'_k)$ entstehe aus dem Tupel $(w_1, \ldots, w_{m+1})$, indem für $j = 1, \ldots, m + 1$ alle Elemente w_j aus $(w_1, \ldots, w_{m+1})$ entfernt werden, für die $w_j = \varepsilon$ gilt.

Der wesentliche Unterschied zwischen den Funktionen Split und Split^+ besteht also darin, dass das von Split erzeugte Tupel auch leere Zeichenketten enthalten kann, während diese von Split^+ aussortiert werden.

3.2.4. Ersetzung von Zeichen in Zeichenketten

Eine weitere wichtige Operation ist die Ersetzung bestimmter Zeichen innerhalb einer Zeichenkette. Für eine Zeichenkette $s := (s_1, \ldots, s_n) \in \mathcal{A}^*$ und zwei Zeichen $c_1, c_2 \in \mathcal{A}$ lässt sich eine entsprechende Funktion Replace : $\mathcal{A}^* \times \mathcal{A} \times \mathcal{A} \to \mathcal{A}^*$ definieren, die alle Vorkommen des Zeichens c_1 innerhalb der Zeichenkette s durch das Zeichen c_2 ersetzt. Diese Funktion ist gegeben durch $\text{Replace}(s, c_1, c_2) := (s'_1, \ldots, s'_n)$ mit

$$s'_i := \begin{cases} c_2, & \text{falls } s_i = c_1 \\ s_i, & \text{sonst.} \end{cases}$$

Falls für $i = 1, \ldots, n$ an der i-ten Stelle in der Zeichenkette s also das Zeichen c_1 steht, wird dieses jeweils durch c_2 ersetzt. Andernfalls wird das entsprechende Zeichen nicht verändert.

Darauf aufbauend wird zusätzlich eine zweite Funktion ReplaceAll definiert als ReplaceAll : $\mathcal{A}^* \times \mathcal{P}(\mathcal{A}) \times \mathcal{A} \to \mathcal{A}^*$. Hierbei bezeichne $\mathcal{P}(\mathcal{A})$ die Potenzmenge von $\mathcal{A}$, also die Menge aller Teilmengen von $\mathcal{A}$. Weiterhin sei $s := (s_1, \ldots, s_n) \in \mathcal{A}^*$ erneut eine Zeichenkette, $C \in \mathcal{P}(\mathcal{A})$ eine Teilmenge von $\mathcal{A}$ und $c \in \mathcal{A}$ ein Zeichen mit $c \notin C$. Dann ist $\text{ReplaceAll}(s, C, c) := (s'_1, \ldots, s'_n)$ mit

$$s'_i := \begin{cases} c, & \text{falls } s_i \in C \\ s_i, & \text{sonst.} \end{cases}$$

Während also die Funktion Replace alle Vorkommen eines einzelnen Zeichens in einer Zeichenkette ersetzt, ersetzt die Funktion ReplaceAll alle Vorkommen aller Zeichen aus einer gegebenen Menge in einer Zeichenkette.

3.2.5. Entfernung von Zeichen aus Zeichenketten

Gelegentlich müssen bestimmte Zeichen aus einer gegebenen Zeichenkette entfernt werden. Hierzu seien $s \in \mathcal{A}^*$ eine Zeichenkette und $c \in \mathcal{A}$ ein Zeichen. Dann lässt sich

eine Abbildung Remove : $\mathcal{A}^* \times \mathcal{A} \to \mathcal{A}^*$ definieren als Remove$(s, c) := (s'_1, \ldots, s'_n)$ mit

$$s'_i := \begin{cases} \varepsilon, & \text{falls } s_i = c \\ s_i, & \text{sonst.} \end{cases}$$

Die Funktion Remove ersetzt also alle Vorkommen des angegebenen Zeichens c in der Zeichenkette s durch die leere Zeichenkette ε und entfernt damit c aus s.

Um alle Vorkommen aller Zeichen einer gegebenen Menge von Zeichen $C \subset \mathcal{A}$ aus einer Zeichenkette s zu entfernen, wird zusätzlich noch die Funktion RemoveAll : $\mathcal{A}^* \times \mathcal{P}(\mathcal{A}) \to \mathcal{A}^*$ definiert als RemoveAll$(s, C) := (s'_1, \ldots, s'_n)$ mit

$$s'_i := \begin{cases} \varepsilon, & \text{falls } s_i \in C \\ s_i, & \text{sonst.} \end{cases}$$

Während also Remove alle Vorkommen eines einzelnen Zeichens aus einer Zeichenkette entfernt, löscht RemoveAll alle Vorkommen aller Zeichen aus einer gegebenen Zeichenmenge.

3.2.6. Umwandlung von Zeichenketten in Kleinbuchstaben

In nachfolgenden Kapiteln wird deutlich werden, dass es sinnvoll sein kann, alle Buchstaben einer Zeichenkette einheitlich in Kleinschreibweise zu überführen, bevor weitere Operationen auf dieser ausgeführt werden.

Als Hilfsfunktion sei hierzu zunächst die Abbildung Unicode : $\mathcal{A} \to \{1, \ldots, 136\,755\} \subset \mathbb{N}$ definiert, die jedem Zeichen $c \in \mathcal{A}$ dessen zugehörige dezimale Unicode-Repräsen-

tation zuweist. Der Unicode-Standard [232] definiert 136 755 Zeichen und es ist

$$\begin{aligned}
\text{Unicode}(\text{„A“}) &= 65,\\
\text{Unicode}(\text{„B“}) &= 66,\\
&\dots\\
\text{Unicode}(\text{„Z“}) &= 90,\\
\text{Unicode}(\text{„a“}) &= 97,\\
&\dots\\
\text{Unicode}(\text{„z“}) &= 122,\\
\text{Unicode}(\text{„Ä“}) &= 196,\\
\text{Unicode}(\text{„Ö“}) &= 214,\\
\text{Unicode}(\text{„Ü“}) &= 220,\\
\text{Unicode}(\text{„ä“}) &= 228,\\
\text{Unicode}(\text{„ö“}) &= 246,\\
\text{Unicode}(\text{„ü“}) &= 252.
\end{aligned}$$

Es wird deutlich, dass zwischen Großbuchstaben und ihren zugehörigen Kleinbuchstaben jeweils eine Differenz von 32 liegt.

Weil die Funktion Unicode eine Bijektion ist, die alle Elemente in $\mathcal{A}$ fortlaufend nummeriert und jedem Element eine eindeutige Zahl zwischen 1 und 136 755 zuweist, existiert eine eindeutig bestimmte inverse Abbildung $\text{Unicode}^{-1} : \{1, \dots, 136\,755\} \to \mathcal{A}$, die einer gegebenen Zahl ihr entsprechendes Unicode-Zeichen zuordnet [62].

Sei $s \in \mathcal{A}^*$ eine Zeichenkette. Dann lässt sich eine Abbildung $\text{ToLower} : \mathcal{A}^* \to \mathcal{A}^*$

definieren als $\text{ToLower}(s) := (s'_1, \ldots, s'_n)$ mit

$$s'_i := \begin{cases} \text{Unicode}^{-1}(\text{Unicode}(s_i) + 32), & \text{falls } 65 \leq \text{Unicode}(s_i) \leq 90 \\ & \vee\ \text{Unicode}(s_i) = 196 \\ & \vee\ \text{Unicode}(s_i) = 214 \\ & \vee\ \text{Unicode}(s_i) = 220 \\ s_i, & \text{sonst.} \end{cases}$$

Die Funktion ToLower wandelt also alle Standardzeichen einer Zeichenkette in Kleinbuchstaben um, indem sie zur dezimalen Unicode-Repräsentation des Zeichens einen Offset-Wert von 32 addiert, wenn es sich um einen Großbuchstaben „A“ bis „Z“ oder einen Umlaut „Ä“, „Ö“ oder „Ü“ handelt. Sonstige Zeichen werden nicht modifiziert.

3.2.7. Bestimmung der Ähnlichkeit zwischen Wörtern

Zur Bestimmung der Ähnlichkeit zwischen zwei Zeichenketten wird die sogenannte *Levenshtein-Distanz* [143] verwendet, die auf den berühmten russischen Mathematiker Vladimir Levenshtein zurückgeht. Diese Metrik, die im Englischen gelegentlich auch als „Edit Distance“ bezeichnet wird, ist eine wohlbekannte und häufig genutzte Metrik zur Messung der Unterschiede zwischen zwei Zeichenketten. Sie berechnet die minimale Anzahl an Einfüge-, Lösch- und Ersetzungsoperationen, die notwendig sind, um eine Zeichenkette in eine andere zu überführen [243].

Für zwei Zeichenketten $a, b \in \mathcal{A}^*$ mit den jeweiligen Längen $i := |a|$ und $j := |b|$ sei die rekursive Funktion $r_{a,b} : \mathbb{N}_0^2 \to \mathbb{N}_0$ gegeben durch

$$r_{a,b}(i,j) := \begin{cases} \max(i,j), & \text{falls } \min(i,j) = 0, \\ \min \begin{cases} r_{a,b}(i-1,j) + 1 \\ r_{a,b}(i,j-1) + 1 \\ r_{a,b}(i-1,j-1) + \chi(a_i, b_j) \end{cases} & \text{sonst,} \end{cases}$$

wobei $\chi(a_i, b_j) : \mathcal{A} \times \mathcal{A} \to \{0, 1\}$ die charakteristische Funktion bezeichne [99]. Diese sei definiert durch

$$\chi(a_i, b_j) := \begin{cases} 0, & \text{falls } a_i = b_j, \\ 1, & \text{falls } a_i \neq b_j. \end{cases}$$

Dann kann die Levenshtein-Distanz $\text{lev}(a, b) : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{N}_0$ definiert werden als $\text{lev}(a, b) := r_{a,b}(|a|, |b|)$.

Die Levenshtein-Distanz zweier unterschiedlicher Zeichenketten ist mindestens so groß wie die Differenz ihrer Länge. Darüber hinaus ist sie höchstes so groß wie die längere der beiden Zeichenketten. Sie ist genau dann 0, wenn die Zeichenketten identisch sind. Insbesondere gilt für die Levenshtein-Distanz die Dreiecksungleichung, d. h. die Levenshtein-Distanz zwischen zwei Zeichenketten ist immer kleiner gleich der Summe ihrer Levenshtein-Distanzen zu einer dritten Zeichenkette. Für drei Zeichenketten $a, b, c \in \mathcal{A}^*$ gilt also $\text{lev}(a, b) \leq \text{lev}(a, c) + \text{lev}(b, c)$.

Mit Hilfe der Levenshtein-Distanz kann nun die Ähnlichkeit zweier Wörter bestimmt werden. Hierzu wird eine Formel verwendet, die auf dem Ansatz von Soukoreff and MacKenzie [215] basiert. Die Ähnlichkeit der Wörter $w_1, w_2 \in \mathcal{A}^*$ lässt sich demzufolge berechnen durch eine Funktion $\sigma : \mathcal{A}^* \times \mathcal{A}^* \to [0, 1] \subset \mathbb{R}$ mit

$$\sigma(w_1, w_2) := 1 - \frac{\text{lev}(w_1, w_2)}{\max(|w_1|, |w_2|)}.$$

Mit den vorangegangenen Erläuterungen zu den Eigenschaften der Levenshtein-Distanz lässt sich leicht nachvollziehen, dass die resultierenden Werte stets zwischen 0 und 1 liegen. Ein Wert von 1 zeigt dabei eine exakte Übereinstimmung an, d. h. die Wörter sind identisch, während ein Wert von 0 anzeigt, dass die Wörter komplett verschieden sind. Je ähnlicher die Wörter w_1 und w_2 sind, desto näher liegt $\sigma(w_1, w_2)$ bei 1.

Beispiel Es folgt ein kurzes Beispiel zur Illustration des zuvor eingeführten Ähnlichkeitsmaßes. Seien hierzu zwei Wörter $w_1, w_2 \in \mathcal{A}^*$ gegeben mit $w_1 :=$ „Tier“ und $w_2 :=$ „Tor“. Die Levenshtein-Distanz dieser beiden Wörter beträgt 2, weil zwei

Operationen notwendig sind, um die eine Zeichenkette in die andere zu überführen. Eine mögliche Folge von Operationen zur Transformation der Zeichenkette „Tier“ in die Zeichenkette „Tor“ wäre:

1. Ersetze das Zeichen „i“ in „Tier“ durch „o“. Dies führt zur Zeichenkette „Toer“.
2. Lösche das Zeichen „e“ aus „Toer“. Dies führt zur Zeichenkette „Tor“.

Es ist somit lev(„Tier“, „Tor“) = 2. Umgekehrt könnte man natürlich mit der gleichen Anzahl an Operationen auch die Zeichenkette „Tor“ in die Zeichenkette „Tier“ überführen.

Entsprechend gilt für die Länge der beiden Zeichenketten, dass |„Tier“| = 4 und |„Tor“| = 3 ist. Hieraus folgt insgesamt als Ähnlichkeit für die beiden Zeichenketten

$$\sigma(\text{„Tier“}, \text{„Tor“}) = 1 - \frac{\text{lev}(\text{„Tier“}, \text{„Tor“})}{\max(|\text{„Tier“}|, |\text{„Tor“}|)} = 1 - \frac{2}{4} = 0{,}5.$$

3.2.8. Bestimmung der Ähnlichkeit zwischen Dokumenten

Während mit der Levenshtein-Distanz die Ähnlichkeit zwischen zwei Wörtern bestimmt werden kann, wird zur Bestimmung der Ähnlichkeit zwischen zwei Texten (die in der Regel jeweils aus mehreren Wörtern bestehen) die sogenannte *Kosinus-Ähnlichkeit* verwendet. Diese ist ein weit verbreitetes Maß zur Bestimmung von Ähnlichkeiten im Bereich des Information Retrievals und Text Minings [69].

Die betrachteten Texte können dabei jeweils aus beliebig vielen Wörtern bestehen, die in der Regel durch Leerzeichen voneinander getrennt sind. Im Information Retrieval werden diese Texte auch als *Dokumente* bezeichnet. Dokumente werden in diesem Kontext zumeist als Tupel von Wörtern notiert, also als Elemente aus $\mathcal{A}^{*^m}$.

Die Grundidee besteht darin, ein Dokument durch einen numerischen Vektor zu charakterisieren. Jedes Element dieses Vektors zeigt an, wie häufig ein bestimmtes Wort in dem entsprechenden Dokument vorkommt. Zwei Dokumente können als ähnlich angesehen werden, wenn sie die gleichen Wörter in ähnlicher Häufigkeit

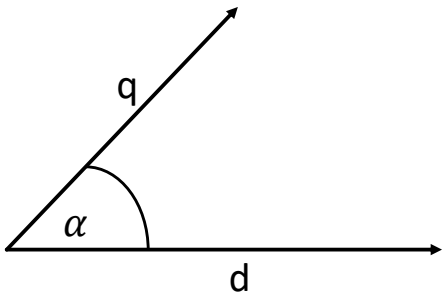

Abbildung 3.2.: Der Winkel α zwischen zwei Vektoren q und d ist kleiner, je ähnlicher die Richtung ist, in die q und d laufen.

enthalten. Je ähnlicher also zwei Dokumente sind, desto ähnlicher sind folglich die Elemente in den charakterisierenden Vektoren.

Man überlegt sich nun, dass Vektoren, die ähnlich zueinander sind, in eine ähnliche Richtung laufen. Weiterhin ist der Winkel zwischen zwei Vektoren kleiner, je ähnlicher die Richtung ist, in die sie laufen. Abbildung 3.2 illustriert diesen Sachverhalt.

Der Kosinus des eingeschlossenen Winkels zwischen zwei Vektoren $q, d \in \mathbb{R}^n$ kann mit Hilfe der folgenden geometrischen Formel bestimmt werden [64]:

$$q \cdot d = |q| \cdot |d| \cdot \cos(\alpha).$$

Durch Umstellen dieser Formel nach $\cos(\alpha)$ kann man nun die Kosinus-Ähnlichkeit als Abbildung $\mathrm{CosSim} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ wie folgt bestimmen:

$$\mathrm{CosSim}(q, d) := \cos(\alpha) = \frac{q \cdot d}{|q| \cdot |d|} = \frac{\sum_{i=1}^{n} q_i \cdot d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \cdot \sqrt{\sum_{i=1}^{n} d_i^2}}.$$

Die Kosinus-Ähnlichkeit zweier Vektoren entspricht somit gerade dem Kosinus des eingeschlossenen Winkels und ist damit ein nützliches Maß dafür, wie ähnlich zwei Dokumente hinsichtlich ihrer textuellen Inhalte sind [213].

Die Werte für die Kosinus-Ähnlichkeit liegen im Intervall $[-1, 1]$ (vgl. Winkeltabelle A.1 sowie zugehörige Abbildung A.1 im Anhang), wobei -1 anzeigt, dass die Vektoren genau entgegengerichtet verlaufen, während 1 anzeigt, dass sie genau gleichgerichtet verlaufen. Ein Wert von 0 zeigt üblicherweise an, dass die Vektoren unabhängig sind und orthogonal zueinander stehen.

Im Kontext des Information Retrievals stellen die Vektoren q und d wie eingangs beschrieben Häufigkeitsvektoren dar. Sie enthalten somit keine reellen Zahlen sondern Elemente aus $\mathbb{N}_0$. Weil das Produkt der Vektoren somit niemals negativ sein kann, liegt die Kosinus-Ähnlichkeit in diesem Fall stets zwischen 0 und 1.

Um für zwei Dokumente $s \in \mathcal{A}^{*^l}$ und $u \in \mathcal{A}^{*^m}$ die zugehörigen Häufigkeitsvektoren q und d zu bestimmen, kann man wie folgt vorgehen:

Zunächst bestimmt man die zugrundeliegende Menge W aller Wörter, die in s oder u vorkommen, durch

$$W := \{w \in \mathcal{A}^* \mid w \in s \ \vee \ w \in u\}.$$

Es sei $k := |W|$ die Anzahl der Elemente in W.

W ist eine endliche Menge mit k Elementen. Deshalb existiert eine Bijektion $f : W \to \{1, \ldots, k\} \subset \mathbb{N}$, die alle Elemente in W fortlaufend nummeriert und jedem Element eine eindeutige Zahl zwischen 1 und k zuweist [62].

Es sei f^{-1} die inverse Abbildung zu f. Diese existiert und ist eindeutig bestimmt, weil f eine bijektive Abbildung ist [62].

Damit kann man die Vektoren q und d wie folgt definieren:

Es sei $q := (q_1, \ldots, q_k) \in \mathbb{N}_0^k$. Für $1 \leq t \leq k$ sei q_t die Häufigkeit, mit der das Wort $f^{-1}(t) \in W$ in s vorkommt.

Weiterhin sei $d := (d_1, \ldots, d_k) \in \mathbb{N}_0^k$. Für $1 \leq t \leq k$ sei d_t die Häufigkeit, mit der das Wort $f^{-1}(t) \in W$ in u vorkommt.

Abschließend kann man die Kosinus-Ähnlichkeit zwischen q und d wie zuvor beschrieben durch $\text{CosSim}(q,d) = \frac{q \cdot d}{|q| \cdot |d|}$ berechnen.

Wenn q der Häufigkeitsvektor für s und d der Häufigkeitsvektor für u ist, so wird abkürzend hierfür im Folgenden auch $\text{CosSim}_{s,u} := \text{CosSim}(q,d)$ geschrieben.

Beispiel Es folgt ein kurzes Beispiel zur Illustration der Kosinus-Ähnlichkeit. Seien hierzu $s :=$ „Das erste Dokument“ und $u :=$ „Das zweite Dokument“. Dann ist die zugrundeliegende Menge W aller Wörter gegeben durch

$$W := \{\text{„Das“}, \text{„erste“}, \text{„Dokument“}, \text{„zweite“}\}.$$

Um die Häufigkeitsvektoren q und d zu bestimmen ist nun für jedes Element aus W zu betrachten, wie oft es in den entsprechenden Dokumenten s und u vorkommt.

Die Wörter „Das“, „erste“ und „Dokument“ kommen jeweils einmal in s vor. Das Wort „zweite“ kommt hingegen nicht vor. Der erste Häufigkeitsvektor ergibt sich somit zu $q = (1,1,1,0)$.

Entsprechend kommt das Wort „Das“ einmal in u vor, während das Wort „erste“ dort nicht vorkommt. Weiterhin kommen die Wörter „Dokument“ und „zweite“ jeweils einmal in u vor. Der zweite Häufigkeitsvektor ergibt sich folglich zu $d = (1,0,1,1)$.

Mit Hilfe der Kosinus-Ähnlichkeit lässt sich nun die Ähnlichkeit zwischen den Dokumenten s und u bestimmen durch

$$\text{CosSim}((1,1,1,0),(1,0,1,1)) = \frac{(1,1,1,0) \cdot (1,0,1,1)}{|(1,1,1,0)| \cdot |(1,0,1,1)|} = \frac{2}{\sqrt{3} \cdot \sqrt{3}} = 0{,}6667.$$

Abkürzend kann dies als

$$\text{CosSim}_{s,u} = 0{,}6667$$

geschrieben werden.

3.2.9. Deskriptive Statistik

In den nachfolgenden Kapiteln werden an verschiedenen Stellen empirische Daten durch Kennzahlen der deskriptiven Statistik dargestellt.

Hierbei bezeichnen die Abkürzungen *Min* und *Max* jeweils das Minimum bzw. Maximum der betrachteten Daten.

Der Mittelwert der Daten wird mit $\overline{x}$ bezeichnet. Für eine Stichprobe $x_1, \ldots, x_n$ mit n Elementen ist dieser gegeben durch

$$\overline{x} := \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Die (empirische) Standardabweichung der Daten wird mit s bezeichnet. Sie gibt die Streuung einer Stichprobe um das arithmetische Mittel an und berechnet sich gemäß der gängigen Formel [21]

$$s := \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}.$$

3.3. Datenbasis für Kalibrierung und Evaluation

Sowohl zur Kalibrierung der in diesem Kapitel diskutierten Lösungsstrategien als auch zur Evaluation des erarbeiteten Konzepts werden reale Interaction-Room-Projekte betrachtet, die auf den digitalen Whiteboards im AugIR rekonstruiert wurden. Hierdurch steht eine realistische Datenbasis zur Verfügung und es wird die Wahrscheinlichkeit erhöht, dass die ermittelten Ergebnisse auf zukünftige Projekte übertragbar sind.

Die Interaction-Room-Methode wurde bereits vielfach in verschiedenen Industrieprojekten eingesetzt [26, 28, 84, 85]. Die meisten dieser Projekte sind gut dokumentiert und die entsprechenden Projektdokumentationen enthalten häufig detaillierte Fotos

der erstellten Skizzen, Mitschriften der Diskussionen sowie eine Übersicht über die platzierten Annotationen.

Aus den bisher durchgeführten Interaction-Room-Projekten wurden 16 besonders gut dokumentierte ausgewählt. Anhand von Aufzeichnungen und Fotoprotokollen wurden sie von fünf verschiedenen Personen auf digitalen Whiteboards nachgezeichnet, um eine möglichst große Anzahl unterschiedlicher Handschriften zu berücksichtigen. Dies soll reale Workshop-Szenarien emulieren, in denen die Skizzen typischerweise ebenfalls von verschiedenen Stakeholdern gezeichnet werden.

Die ausgewählten Projekte sind heterogen und stammen aus unterschiedlichen Domänen: Vier Projekte wurden mit Unternehmen aus der Finanzbranche durchgeführt, sechs Projekte mit Versicherungsunternehmen, zwei mit IT-Service-Dienstleistern, zwei mit internationalen Großhändlern, eins mit einer Lotto-Gesellschaft und eins mit einem Marktforschungsunternehmen.

Die Anzahl der in jedem Projekt erstellten Skizzen korreliert in der Regel mit der Anzahl und Dauer der in diesem Projekt durchgeführten Interaction-Room-Workshops. Diese variierten zwischen kurzen zweitägigen Workshops und mehreren zusammenhängenden Sitzungen, die sich über einen Zeitraum von mehreren Wochen erstreckten.

Die Anzahl der teilnehmenden Stakeholder variierte ebenfalls stark. Es gab sowohl einzelne Workshops mit sieben Teilnehmern als auch aufeinander aufbauende Workshops mit insgesamt mehr als zwanzig wechselnden Stakeholdern.

Tabelle 3.1 zeigt eine Übersicht über die Skizzen aller 16 rekonstruierten Projekte. Aus Gründen des Datenschutzes wurden alle Projekte anonymisiert. Zur eindeutigen Identifikation werden im Folgenden die Bezeichnungen $P_1, \ldots, P_{16}$ verwendet, die in der ersten Spalte der Tabelle aufgeführt sind.

Die zweite Spalte zeigt die Sprache des jeweiligen Projekts und seiner entsprechenden Skizzen. Projekt P_1 wurde mit internationalen Stakeholdern in Englisch durchgeführt und enthält folglich ausschließlich englische Texte. Alle anderen 15 Projekte enthalten nur Texte in deutscher Sprache.

Tabelle 3.1.: Übersicht über die Skizzen aller 16 digital rekonstruierten Interaction-Room-Projekte.

		Anzahl der Skizzen				
Projekt	Sprache	Feature Canvas	Process Canvas	Object Canvas	Integration Canvas	Gesamt
P_1	EN	0	1	1	2	4
P_2	DE	1	1	1	0	3
P_3	DE	1	2	0	0	3
P_4	DE	0	1	3	1	5
P_5	DE	1	4	1	0	6
P_6	DE	0	1	1	1	3
P_7	DE	1	5	0	0	6
P_8	DE	1	1	2	0	4
P_9	DE	1	2	1	0	4
P_{10}	DE	1	3	2	0	6
P_{11}	DE	1	1	1	0	3
P_{12}	DE	1	6	1	0	8
P_{13}	DE	1	4	0	0	5
P_{14}	DE	0	5	3	2	10
P_{15}	DE	1	4	0	0	5
P_{16}	DE	1	12	0	1	14
	$\sum$	12	53	17	7	89
	Ø	0,75	3,31	1,06	0,44	5,56

Die Spalten drei bis sechs führen auf, wie viele Skizzen eines Typs im jeweiligen Projekt erstellt wurden. Spalte sieben zeigt die Gesamtzahl der in einem Projekt erstellten Skizzen als Summe über alle vorgenannten Skizzen-Typen.

Unterhalb der Tabelle sind für jede Spalte sowohl die Summe der Einzelwerte als auch der durchschnittliche Wert pro Projekt angegeben.

Insgesamt umfassen alle Projekte zusammen 89 Canvases, wobei durchschnittlich zwischen 5 und 6 Canvases pro Projekt erstellt wurden. Die gezeigten Zahlen unterstreichen dabei die Heterogenität der betrachteten Projekte: Es sind sowohl kleinere Projekte mit wenigen Skizzen enthalten, wie beispielsweise Projekt P_2 mit lediglich insgesamt drei Skizzen, als auch größere Projekte, wie beispielsweise Projekt P_{16}, welches 14 Skizzen umfasst.

Erwartungsgemäß unterscheiden sich auch die verwendeten Canvas-Typen. Diese richten sich in der Regel nach dem jeweiligen Projektkontext. Während einige Projekte auf dynamische Aspekte und Prozesse fokussierten und deshalb viele Process Canvases enthalten, haben sich Stakeholder in anderen Projekten stattdessen eher auf Objekte und Schnittstellen konzentriert und deshalb überwiegend Object und Integration Canvases erstellt.

Wie in Abschnitt 1.2 beschrieben, beginnt ein idealtypischer Interaction-Room-Workshop meistens mit der Erfassung von Anforderungen, die in Form von Story Cards auf einem Feature Canvas festgehalten werden. Deshalb enthalten dreiviertel aller rekonstruierten Projekte einen Feature Canvas.

In der Regel werden im Verlauf eines Workshops mehrere Process Canvases erstellt. Die durchschnittliche Anzahl der Process Canvases pro Projekt beläuft sich auf 3,31. Im Durchschnitt erstellten die Stakeholder also in jedem Projekt etwa drei bis vier Skizzen dieses Typs.

Nur in seltenen Fällen werden in einem Interaction-Room-Workshop mehrere Object Canvases erstellt. Für gewöhnlich werden alle fachlichen Objekte in einem gemeinsamen Canvas gesammelt und zueinander in Beziehung gesetzt. Dies spiegelt sich in der durchschnittlichen Anzahl von 1,06 Object Canvases pro Projekt wieder.

Integration Canvases kommen nicht in jedem Projekt zum Einsatz, weil eine Betrachtung der umgebenden Systeme und die Analyse der Kommunikation mit Komponenten der Systemlandschaft nicht immer im Fokus des Interesses stehen. Man sieht, dass nur in 5 von 16 Projekten die Integration des betrachteten Projekts in die Anwendungs- und Systemlandschaft Bestandteil des Workshops war. Die durchschnittliche Anzahl der Integration Canvases pro Projekt liegt deshalb lediglich bei 0,44.

Darüber hinaus fällt auf, dass keines der vorgestellten Projekte einen Interaction Canvas enthält. In allen betrachteten Projekten lag der Fokus auf der Entwicklung und Diskussion klassischer Informationssysteme, die häufig nicht über komplexe grafische Benutzerschnittstellen verfügen.

Wie bereits in Abschnitt 1.2 erläutert, besitzt die Interaction-Room-Methode einen modularen Aufbau und stellt bewusst eine Auswahl verschiedener Perspektiven zur Verfügung, um für eine möglichst große Zahl unterschiedlicher Projekte und Domänen adaptiert werden zu können. Deshalb wird ein einzelnes Projekt in der Praxis nur selten alle Canvas-Typen verwenden, die durch die Interaction-Room-Methode definiert sind.

Tabelle 3.2 zeigt einen Überblick über die Inhalte der Skizzen aller 16 rekonstruierten Projekte. Spalte eins gibt erneut die eindeutigen Projektkürzel $P_1, \ldots, P_{16}$ an. In den Spalten zwei bis fünf ist daneben aufgeführt, wie viele Textelemente in den Skizzen eines jeweiligen Typs enthalten sind.

Als *Textelemente* werden an dieser Stelle alle handgezeichneten Elemente bezeichnet, die aus einer logisch zusammengehörigen Menge von Freihandlinienzügen bestehen und vom Handschrifterkenner in sinnvollen Text übersetzt werden konnten, der jeweils ein oder mehrere Wörter umfasst. Hierbei handelt es sich beispielsweise um die Inhalte der Story Cards in Feature Canvases, um die Beschriftungen der Aktionen in Process Canvases sowie um sonstige handschriftliche Notizen und Anmerkungen in den Skizzen.

Spalte sechs listet die Summe aller in einem Projekt vorkommenden Textelemente über alle Skizzen auf. In den letzten beiden Zeilen der Tabelle sind erneut die Summe

Tabelle 3.2.: Übersicht über enthaltene Textelemente in den Skizzen aller 16 digital rekonstruierten Interaction-Room-Projekte.

Projekt	Anzahl der Textelemente				
	Feature Canvas	Process Canvas	Object Canvas	Integration Canvas	Gesamt
P_1	0	29	24	35	88
P_2	11	12	28	0	51
P_3	19	70	0	0	89
P_4	0	10	30	4	44
P_5	31	44	20	0	95
P_6	0	59	21	29	109
P_7	17	91	0	0	108
P_8	84	30	50	0	164
P_9	40	42	34	0	116
P_{10}	29	32	40	0	101
P_{11}	17	42	14	0	73
P_{12}	39	67	30	0	136
P_{13}	62	96	0	0	158
P_{14}	0	57	38	81	176
P_{15}	29	51	0	0	80
P_{16}	64	264	0	86	414
$\sum$	442	996	329	235	2 002
Ø	27,63	62,25	20,56	14,69	125,13

über alle Einzelwerte einer Spalte sowie der durchschnittliche Wert pro Projekt angegeben.

Rein geometrische Elemente und Freihandformen ohne Text wie beispielsweise Pfeile, Rauten oder Start-/Endknoten von Prozessen wurden in dieser Statistik nicht berücksichtigt, weil sie für die Mehrheit der nachfolgenden Experimente nicht von Bedeutung sind. Dort, wo geometrische Elemente ohne Text eine Rolle spielen, werden diese Zahlen im Folgenden jeweils explizit an der entsprechenden Stelle genannt.

Die rekonstruierten Projekte umfassen 89 Canvases mit insgesamt 2 002 Textelementen. Hierbei entfallen im Durchschnitt 125,13 Textelemente auf jedes Projekt. Man sieht, dass nicht nur Typ und Anzahl der in einem Workshop erstellten Skizzen variieren, sondern auch die Anzahl der darin geschriebenen Texte. Kleine Projekte wie beispielsweise P_4 enthalten lediglich 44 Textelemente, während große Projekte wie beispielsweise P_{16} mehr als 400 Textelemente umfassen können.

Process Canvases enthalten mit durchschnittlich 62,25 Textelementen von allen Canvases den meisten Text. Danach kommen die Feature Canvases mit durchschnittlich 27,63 Textelementen. Weil auf einem Feature Canvas in der Regel jede Story Card einem Textelement entspricht, enthält somit jedes Projekt im Durchschnitt 27 bis 28 Feature-Karten. Object Canvases enthalten durchschnittlich 20,56 Textelemente, wobei jedes Textelement für gewöhnlich einem Fachobjekt entspricht. Integration Canvases enthalten mit 14,69 Texten im Durchschnitt die wenigsten Beschriftungen.

Die vorgestellten Projekte werden im Folgenden für verschiedene Experimente und Studien herangezogen, die zur Kalibrierung der Lösungsstrategien sowie zur Evaluation des erarbeiteten Konzepts dienen.

3.4. Klassifizierung von Linienzügen

Stakeholder erstellen auf den Canvases im Interaction Room Freihandskizzen, die sowohl Texte als auch Formen (wie beispielsweise Pfeile, Rauten und Rechtecke)

enthalten. Damit die Inhalte dieser Skizzen interpretiert und nutzbar gemacht werden können, ist zunächst eine Unterscheidung der gezeichneten Linienzüge in Text und Form unabdingbar.

Die nachfolgenden Abschnitte präsentieren einen Lösungsansatz zur entsprechenden Klassifizierung von Linienzügen. Es wird ein statistisches Verfahren vorgestellt, mit dessen Hilfe Linienzüge analysiert und bewertet werden können. Darüber hinaus werden charakteristische Merkmale von Linienzügen diskutiert, die sich als Klassifikationskriterien eignen.

3.4.1. Einleitung

In Abschnitt 2.3 wurden bereits zahlreiche Arbeiten vorgestellt, die sich mit der Klassifizierung von handgezeichneten Linienzügen beschäftigen. Es wurde deutlich, dass es sich um ein nicht-triviales und grundsätzlich schwer zu lösendes Problem handelt [238]. Die große Anzahl aktueller Forschungsarbeiten, die sich diesem Problem widmen, macht deutlich, dass es bisher noch keine allgemeingültige Lösung gibt, die für alle denkbaren Anwendungsbereiche zufriedenstellend funktioniert [54].

Obwohl die vorgestellten Ansätze teils sehr unterschiedliche Methoden verwenden, liegt fast allen die gleiche Idee zugrunde. Diese besteht darin, dass Text- und Form-Linienzüge in der Regel unterschiedliche charakteristische Eigenschaften aufweisen. Während Texte beispielsweise häufig aus einer schnell gezeichneten Abfolge verhältnismäßig kurzer Linienzüge bestehen, sind viele Formen häufig größer und aus längeren Linienzügen zusammengesetzt. Durch Betrachtung einer ausreichend großen Menge solcher charakteristischen Merkmale wird es möglich, handgezeichnete Linienzüge als potentielle Texte oder Formen zu klassifizieren.

Im Folgenden werden diese charakteristischen Merkmale als *Features (eines Linienzugs)* bezeichnet. Man überlegt sich leicht, dass es nicht ausreichend ist, hierbei nur ein einzelnes Feature zu betrachten: Weil Skizzen im Interaction Room grundsätzlich beliebige handgezeichnete Elemente enthalten können und unterschiedliche Stakeholder jeweils individuelle Handschriften besitzen, treffen auf einen gegebenen Linienzug nur selten alle Eigenschaften mit ihrer vollen Ausprägung zu. Deshalb

kann für gewöhnlich nur durch eine gleichzeitige Betrachtung mehrerer Merkmale hinreichend zuverlässig auf den Typ eines Linienzugs geschlossen werden.

Grundsätzlich lassen sich die möglichen Features jedes Linienzugs in zwei Kategorien einteilen:

Es existieren *lokale Features*, die sich jeweils nur aus den (geometrischen) Eigenschaften eines einzelnen Linienzugs bestimmen. Hierzu zählen beispielsweise die Länge eines Linienzugs und die Anzahl seiner Kontrollpunkte. Weitere auf der Skizze befindliche Linienzüge sind für die Bestimmung der lokalen Merkmale nicht relevant.

Darüber hinaus existieren weiterhin *globale Features*, bei denen die räumliche und zeitliche Nähe eines betrachteten Linienzugs zu benachbarten Linienzügen eine Rolle spielen. Hierzu zählen beispielsweise die Anzahl der Schnittpunkte mit anderen Linienzügen innerhalb der gleichen Skizze oder die zeitliche Differenz zum vorausgegangenen Linienzug.

Der im Folgenden vorgestellte Lösungsansatz basiert darauf, jeden Linienzug durch einen numerischen Vektor zu beschreiben, dessen Elemente die Ausprägungen der betrachteten Features dieses Linienzugs darstellen. Ein solcher Vektor kann dann von einem geeigneten Machine-Learning-Verfahren, welches zuvor auf einer passenden Datenmenge trainiert wurde, analysiert und entsprechend klassifiziert werden.

Hierbei überlegt man sich, dass das bestehende Problem vereinfacht werden kann, indem keine generelle Lösung für beliebige Freihandskizzen angestrebt wird. Stattdessen beschränkt sich der im Folgenden vorgestellte Lösungsansatz explizit auf die Betrachtung von Interaction-Room-Canvases. Obwohl Skizzen, die im Interaction Room erstellt werden, grundsätzlich informell bleiben und keiner strengen Syntax folgen, sind die verschiedenen Canvas-Typen wie in Abschnitt 1.2 beschrieben dennoch an existierende Modellierungssprachen angelehnt. Dies bietet einen Anhaltspunkt für mögliche Inhalte und lässt beispielsweise darauf schließen, dass bestimmte Standard-Formen wie Rauten, Pfeile und Rechtecke häufiger vorkommen als beliebige Freihandformen. Diese Einschränkungen helfen bei der Auswahl geeigneter Features.

Aus diesem Grund wird zur Analyse und Bewertung potentiell geeigneter Klassifizierungskriterien auch keine generische Datenbank mit beliebigen Freihandskizzen wie beispielsweise IAMonDo [112] verwendet. Statt dessen wurden die in Abschnitt 3.3 vorgestellten digital rekonstruierten Interaction-Room-Projekte betrachtet. Diese umfassen 89 verschiedene Skizzen mit insgesamt 48 617 Linienzügen, welche sich in 43 620 Text-Linienzüge und 4 997 Form-Linienzüge unterteilen. Anhand dieser Datenbasis wurden potentielle Features auf ihre Eignung hin untersucht.

Hierbei ist zwischen Texten in Druckschrift und Schreibschrift zu unterscheiden. Während Texte in Druckschrift aus mehreren aufeinanderfolgenden Zeichen bestehen, welche wiederum jeweils aus einem oder mehreren Linienzügen zusammengesetzt sind, bestehen Texte in Schreibschrift häufig aus einem einzigen langen Linienzug.

Der nachfolgend vorgestellte Ansatz konzentriert sich aus mehreren Gründen ausschließlich auf die Betrachtung von Druckschrift: Zum einen lässt sich in der Praxis beobachten, dass die meisten Erwachsenen in Druckschrift schreiben. So liegen beispielsweise alle rekonstruierten Interaction-Room-Projekte ausnahmslos in Druckschrift vor. Weder bei der Erstellung der Canvases auf den analogen Whiteboards noch bei der Beschriftung von Karteikarten für den Feature Canvas wurden Texte in Schreibschrift erstellt. Darüber hinaus würde die Betrachtung von Schreibschrift die Klassifizierung von Linienzügen wesentlich erschweren, weil hierbei die Unterschiede zwischen Texten und Formen verschwimmen. Wenn ein Wort in einem Zug geschrieben und nicht aus einzelnen Buchstaben zusammengesetzt wird, so ist dieser Linienzug zum Beispiel deutlich länger und weist eine höhere Anzahl von Kontrollpunkten auf, was typischerweise eher auf eine Form als auf einen Text hindeutet. Aus diesen Gründen unterstützen auch die meisten verwandten Arbeiten ausschließlich eine Klassifizierung von Texten in Druckschrift.

Abschnitt 3.4.2 stellt zunächst das statistische Verfahren vor, dass zur Auswahl und Bewertung potentieller Merkmale herangezogen wurde. Danach werden in Abschnitt 3.4.3 wichtige Begriffe und Funktionen eingeführt, die für das weitere Verständnis notwendig sind. In den Abschnitten 3.4.4 und 3.4.5 werden daraufhin die ausgewählten lokalen bzw. globalen Features vorgestellt und diskutiert. In Abschnitt 3.4.6

erfolgt eine empirische Bewertung der identifizierten Features und Abschnitt 3.4.7 fasst die gewonnenen Erkenntnisse abschließend kurz zusammen.

3.4.2. Logistische Regression

Die in Abschnitt 2.3 vorgestellten Arbeiten zur Klassifizierung von Linienzügen greifen auf unterschiedliche Techniken wie rekurrente neuronale Netze [111, 181, 238], Entscheidungsbäume [182, 191] und Conditional Random Fields (CRFs) [57, 60, 135] zurück.

Anders als bei diesen Ansätzen wird im Folgenden die sogenannte *logistische Regression* zur Klassifizierung von Linienzügen verwendet. Diese wurde bereits 1958 vom Statistiker David Cox entwickelt [48]. Bevor die Vorteile dieses Verfahrens gegenüber anderen Techniken erläutert werden, folgt zunächst eine kurze Einführung in die Grundlagen der logistischen Regression.

Die Grundidee dieses Verfahrens besteht darin, die Wahrscheinlichkeit für die Ausprägung einer binären Variable, die nur die beiden Werte 1 und 0 annehmen kann, in Abhängigkeit einer oder mehrerer unabhängiger Variablen zu ermitteln. Hierbei kann insbesondere untersucht werden, welchen Einfluss jede betrachtete unabhängige Variable auf die Ausprägung der abhängigen Variable hat und wie stark dieser ist.

Hierzu wird eine Regressionsgleichung der folgenden Form aufgestellt [20]:

$$\text{Logit}(P(Y = 1)) = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k$$

Diese gibt die Wahrscheinlichkeit dafür an, dass die Ausprägung der abhängigen Variable Y den Wert 1 annimmt. Die $X_1, \ldots, X_k$ repräsentieren die unabhängigen Variablen und die $\beta_1, \ldots, \beta_k$ die zugehörigen sogenannten Regressionskoeffizienten. Durch Analyse eines gegebenen Trainingsdatensatzes können diese Koeffizienten bestimmt werden und aus der Größe eines Koeffizienten lässt sich der Einfluss der zugehörigen unabhängigen Variable auf das Ergebnis ableiten.

Zur Ermittlung und Bewertung von Klassifikationskriterien für Linienzüge wird im Folgenden der Typ des betrachteten Linienzugs als abhängige Variable gewählt. Die beiden möglichen Ausprägungen hierfür können als

$$1 = \text{Text-Linienzug}$$

und

$$0 = \text{Form-Linienzug}$$

festgelegt werden. Die zu untersuchenden Features zur Klassifizierung von Linienzügen stellen dementsprechend die unabhängigen Variablen dar.

Zur Auswahl geeigneter Klassifikationskriterien mit Hilfe der logistischen Regression wurde nun wie folgt vorgegangen:

Zunächst wurden die Linienzüge in den zuvor erwähnten rekonstruierten Interaction-Room-Projekten gründlich analysiert, um erste charakteristische Merkmale zu identifizieren. Darüber hinaus wurde eine umfassende Literaturrecherche durchgeführt, um weitere potentielle Kriterien zusammenzutragen [38, 58, 59, 74, 97, 152, 171, 181, 182, 237, 191, 196, 238, 245]. Auf diese Weise konnten insgesamt 89 potentielle Features zur Klassifizierung von Linienzügen ermittelt werden.

Um die tatsächliche Nützlichkeit dieser Merkmale zu bewerten, wurde ein Machine-Learning-System mit logistischer Regression implementiert. Dieses wurde mit allen potentiellen Features trainiert, wodurch die Koeffizienten der Regressionsgleichung bestimmt wurden. Anschließend wurden die Werte der Koeffizienten betrachtet, um zu ermitteln, welchen Einfluss das jeweils zugehörige Feature auf die Ausprägung eines Linienzugs hat.

In mehreren Iterationen wurde nun der Reihe nach jeweils das Feature mit dem kleinsten Einfluss entfernt. Nach der Entfernung jedes Merkmals wurde stets das System neu trainiert und die Qualität der Klassifizierung neu bewertet. Dies wurde solange wiederholt, bis eine Entfernung weiterer Features die Qualität der Klassifizierung maßgeblich negativ beeinträchtigt hätte. Auf diese Weise konnte eine möglichst geringe Anzahl relevanter Merkmale identifiziert werden, die Linienzüge hinreichend genau als Text- oder Form-Linienzüge klassifizieren.

Die logistische Regression bietet gegenüber anderen Ansätzen insbesondere zwei nennenswerte Vorteile:

Zum einen ist ein entsprechendes Machine-Learning-System, das das Verfahren der logistischen Regression implementiert, sehr leicht zu konfigurieren. Es müssen nur wenige Parameter eingestellt werden, damit das System zufriedenstellende Ergebnisse liefert. Demgegenüber verlangen andere Algorithmen, wie beispielsweise die Support Vector Machines, einen deutlich höheren Konfigurationsaufwand, um vergleichbar gute Ergebnisse zu erzielen.

Darüber hinaus unterstützt die logistische Regression wie zuvor erläutert bei der Auswahl passender Merkmale. Durch Analyse der Koeffizienten konnte sehr einfach und schnell der Einfluss jedes Merkmals auf das Gesamtergebnis der Regressionsgleichung ermittelt werden. Hierdurch ließ sich bereits mit wenigen Iterationen eine annähernd minimale Menge geeigneter Klassifikationskriterien bestimmen. Mit anderen Verfahren wäre dies nicht vergleichbar effizient möglich gewesen.

Studien zeigen zwar, dass die Qualität der Klassifizierung von Linienzügen in beliebigen Freihandskizzen mit anderen Verfahren, wie beispielsweise Support Vector Machines, besser ausfallen kann [54], jedoch wird man in Abschnitt 3.4.6 sehen, dass die logistische Regression für Interaction-Room-Skizzen bereits ein zufriedenstellendes Ergebnis liefert.

3.4.3. Definitionen

In diesem Abschnitt werden einige Begriffe und Funktionen definiert, die bei der nachfolgenden Diskussion der Kriterien zur Klassifizierung von Linienzügen Anwendung finden.

Die Zeichenoperation für einen neuen Linienzug beginnt, sobald ein Stakeholder einen Stift auf die Zeichenfläche eines digitalen Whiteboards setzt. Solange er den Stift auf der Zeichenfläche umher bewegt, werden kontinuierlich Abtastpunkte dieser Bewegung erfasst und der Reihe nach in einer geordneten Liste gesammelt. Die

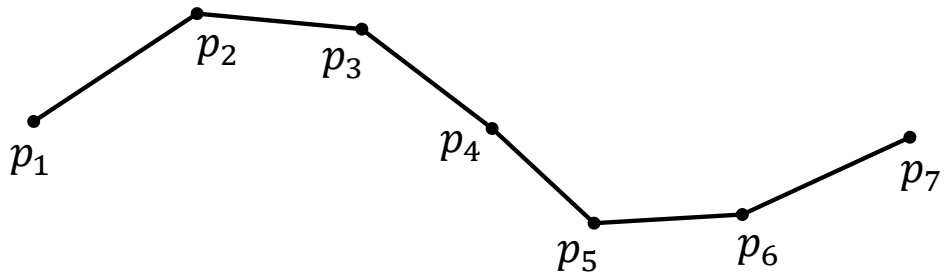

Abbildung 3.3.: Beispiel für einen von links nach rechts gezeichneten Linienzug mit 7 Kontrollpunkten $p_1, \ldots, p_7$.

Zeichenoperation gilt als abgeschlossen – und der entsprechende Linienzug als fertig gezeichnet – sobald der Stakeholder den Stift wieder von der Zeichenfläche abhebt.

Durch die vorgenannte Interaktion entsteht ein neuer *Linienzug* auf dem digitalen Whiteboard. Ein solcher Linienzug L_i wird hierbei durch eine Liste von aufeinanderfolgenden Punkten $p_j := (x, y) \in \mathbb{R}^2$ definiert. Diese Punkte werden im Folgenden als *Kontrollpunkte* des Linienzugs bezeichnet. Ein Linienzug L_i mit n Kontrollpunkten ist somit definiert als ein Tupel $L_i := (p_1, \ldots, p_n)$.

Die Anzahl der Kontrollpunkte eines Linienzugs $L_i := (p_1, \ldots, p_n)$ wird im Folgenden mit $|L_i|$ notiert.

Um auszudrücken, dass ein Punkt p_j ein Kontrollpunkt eines Linienzugs L_i ist, wird $p_j \in L_i$ geschrieben.

Abbildung 3.3 zeigt ein Beispiel eines Linienzugs, der aus 7 Kontrollpunkten besteht und von links nach rechts gezeichnet wurde. Zur grafischen Darstellung eines Linienzugs werden jeweils zwei aufeinanderfolgende Kontrollpunkte durch eine gerade Linie miteinander verbunden. Indem der Abstand zwischen den Kontrollpunkten hinreichend klein gewählt wird, lassen sich durch diese gängige Technik auch annähernd runde Formen erzeugen.

Zur besseren Lesbarkeit werden im Folgenden die x- bzw. y-Koordinate eines Kontrollpunkts p_j abkürzend als x_{p_j} bzw. y_{p_j} notiert.

Zusätzlich verfügt jeder Punkt $p_j \in L_i$ über einen Zeitstempel t_{p_j}. Dieser gibt den exakten Zeitpunkt an, zu dem der entsprechende Punkt als Abtastpunkt der Stiftbewegung auf dem digitalen Whiteboard erfasst wurde. Mit Hilfe dieser Zeitstempel können später verschiedene Eigenschaften eines Linienzugs exakt bestimmt werden, beispielsweise die Dauer der Zeichenoperation oder der zeitliche Abstand zwischen zwei Linienzügen.

Die Menge aller Linienzüge in einer Skizze S wird als geordnetes Tupel von Linienzügen $L(S) := (L_1, \ldots, L_u)$ definiert. Die Linienzüge in dieser Liste sind nach dem Zeitpunkt ihrer Fertigstellung, d. h. nach dem Zeitstempel ihres jeweils letzten Kontrollpunkts, sortiert. Für zwei Linienzüge $L_i := (p_1, \ldots, p_n)$ und $L_j := (q_1, \ldots, q_m)$ mit $i < j$ gilt deshalb stets $t_{p_n} < t_{q_m}$.

Die Anzahl aller Linienzüge in einer Skizze S ist entsprechend gegeben durch $|L(S)|$.

Um auszudrücken, dass ein Linienzug L_i in einer Skizze S vorkommt, wird im Folgenden $L_i \in L(S)$ geschrieben.

Weiterhin werden einige Hilfsfunktionen zur Betrachtung des räumlichen und zeitlichen Abstands zwischen Linienzügen und Kontrollpunkten benötigt:

Für zwei Linienzüge $L_i := (p_1, \ldots, p_n)$ und $L_j := (q_1, \ldots, q_m)$ sowie zwei Kontrollpunkte $p_k \in L_i$ und $q_l \in L_j$ sei eine Funktion $\delta : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ zur Bestimmung der euklidischen Distanz zwischen den beiden Punkten p_k und q_l gegeben durch

$$\delta(p_k, q_l) := \sqrt{(x_{p_k} - x_{q_l})^2 + (y_{p_k} - y_{q_l})^2}.$$

Aufbauend auf der Funktion δ kann eine weitere Funktion $\Delta : L(S) \times L(S) \to \mathbb{R}$ definiert werden, die den (minimalen) euklidischen Abstand zwischen zwei Linienzügen L_i und L_j in der Skizze S bestimmt durch

$$\Delta(L_i, L_j) := \min\{\delta(p_k, q_l) \,|\, p_k \in L_i, q_l \in L_j\}.$$

Darüber hinaus sei entsprechend der zeitliche Abstand zwischen zwei Linienzügen durch eine Funktion $\tau : L(S) \times L(S) \to \mathbb{R}$ definiert als

$$\tau(L_i, L_j) := \min\{|t_{p_k} - t_{q_l}| \,|\, p_k \in L_i, q_l \in L_j\}.$$

Im Folgenden wird an verschiedenen Stellen zudem die sogenannte *Bounding Box* eines Linienzugs betrachtet. Hierbei handelt es sich um das kleinste Rechteck, das diesen Linienzug vollständig enthält [73]. Es werden ausschließlich solche Bounding Boxen betrachtet, deren Kanten an der horizontalen x-Achse und der vertikalen y-Achse ausgerichtet sind. Man spricht deshalb auch von einer „Axis-Aligned Bounding Box“ bzw. kurz „AABB“. Für einen Linienzug L_i wird seine Bounding Box mit $\mathcal{B}(L_i)$ bezeichnet. Die Menge aller Bounding Boxen in der euklidischen Ebene wird im Folgenden mit $\mathcal{B}$ notiert. Es ist also $\mathcal{B}(L_i) \in \mathcal{B}$.

Für einen Linienzug $L_i := (p_1, \ldots, p_n)$ seien

$$\begin{aligned}
\mathcal{B}(L_i)_{x_{min}} &:= \min(x_{p_1}, \ldots, x_{p_n}) \\
\mathcal{B}(L_i)_{x_{max}} &:= \max(x_{p_1}, \ldots, x_{p_n}) \\
\mathcal{B}(L_i)_{y_{min}} &:= \min(y_{p_1}, \ldots, y_{p_n}) \\
\mathcal{B}(L_i)_{y_{max}} &:= \max(y_{p_1}, \ldots, y_{p_n})
\end{aligned}$$

die minimalen und maximalen x- und y-Werte aller Kontrollpunkte von L_i sowie

$$\mathcal{B}(L_i)_{min} := (\mathcal{B}(L_i)_{x_{min}}, \mathcal{B}(L_i)_{y_{min}})$$

und

$$\mathcal{B}(L_i)_{max} := (\mathcal{B}(L_i)_{x_{max}}, \mathcal{B}(L_i)_{y_{max}})$$

die linke untere bzw. rechte obere Ecke der Bounding Box von L_i.

Abbildung 3.4 zeigt ein Beispiel für die Bounding Box eines Linienzugs.

Im Folgenden werden 24 lokale und globale Features zur Klassifizierung von Linienzügen vorgestellt. Für einen gegebenen Linienzug L_i werden diese als Vektor

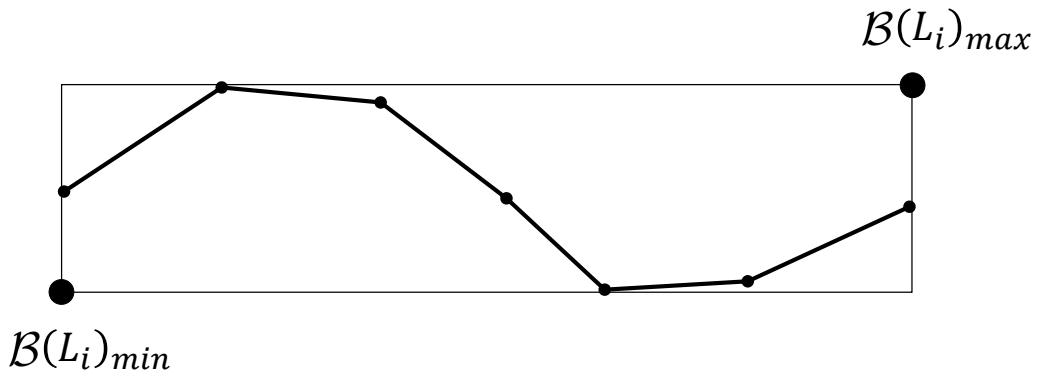

Abbildung 3.4.: Beispiel für die Bounding Box eines Linienzugs L_i.

$f(L_i) := (f_1(L_i), \ldots, f_{24}(L_i)) \in \mathbb{R}^{24}$ spezifiziert. Jedes Element $f_j(L_i) \in f(L_i)$ repräsentiert hierbei den numerischen Wert eines Features des betrachteten Linienzugs.

3.4.4. Lokale Features zur Klassifizierung von Linienzügen

Durch das eingangs beschriebene Verfahren konnten 12 lokale Features zur Klassifizierung von Linienzügen ermittelt werden. Wie zuvor beschrieben bestimmen sich diese jeweils aus den geometrischen Eigenschaften eines einzelnen Linienzugs, ohne dabei die restlichen Linienzüge einer Skizze miteinzubeziehen. Für einen Linienzug L_i bilden sie die ersten 12 Werte $f_1(L_i), \ldots, f_{12}(L_i)$ des charakterisierenden Vektors $f(L_i)$.

Die lokalen Features zur Klassifizierung von Linienzügen werden im Folgenden hergeleitet und mathematisch präzise definiert. Hierzu werden überdies die in Abschnitt 3.2.9 vorgestellten Kennzahlen der deskriptiven Statistik für jedes Merkmal untersucht.

Tabelle 3.3.: Kennzahlen der deskriptiven Statistik für grundlegende Eigenschaften von Linienzügen.

		Länge des Linienzugs in px	Dauer der Zeichenoperation in ms	Anzahl der Kontrollpunkte
Texte	Min	0	0	1
	Max	185,56	985	94
	$\overline{x}$	25,29	309,38	14,14
	s	17,02	156,49	8,51
Formen	Min	0,98	0	2
	Max	2 477,51	10 891	1 240
	$\overline{x}$	214,43	1 372,43	108,7
	s	250,27	1 288,37	125,14

Grundlegende Eigenschaften von Linienzügen

Zunächst überlegt man sich, dass Texte aus Wörtern gebildet werden, die jeweils aus mehreren aneinandergereihten Zeichen aus $\mathcal{A}$ bestehen. Einzelne Zeichen eines Wortes werden in der Regel in rascher Abfolge ohne Unterbrechung direkt hintereinander geschrieben [24, 57, 59]. Dabei besteht jedes Zeichen aus einem oder mehreren kurzen Linienzügen. Somit sind Texte insgesamt aus einer Abfolge vieler kurzer Text-Linienzüge zusammengesetzt.

Demgegenüber sind die meisten Formen häufig aus deutlich weniger Linienzügen aufgebaut. Charakteristisch hierbei ist jedoch, dass viele dieser Form-Linienzüge oft wesentlich länger sind als Text-Linienzüge. Dies trifft insbesondere auf häufig verwendete Standard-Formen wie beispielsweise Pfeile, Assoziationsbeziehungen und Rechtecke zu.

Spalte 1 in Tabelle 3.3 zeigt, dass die Länge von Text-Linienzügen im Durchschnitt 25,29 Pixel beträgt, während sich die Länge von Form-Linienzügen durchschnittlich auf 214,43 Pixel beläuft. Der längste Text-Linienzug aller Projekte weist eine Länge von 185,56 Pixeln auf, während der längste Form-Linienzug eine Länge von 2 477,51 Pixeln besitzt.

Darüber hinaus ist insbesondere auffällig, dass es Text-Linienzüge mit einer minimalen Länge von 0 Pixeln gibt. Hierbei handelt es sich um Punkte, wie beispielsweise über dem Buchstaben „i“. Diese bestehen nur aus einem einzigen Kontrollpunkt und haben deshalb keine messbare Länge.

Während es sich also bei kürzeren Linienzügen in der Regel eher um Texte handelt, repräsentieren längere Linienzüge mit höherer Wahrscheinlichkeit Formen.

Die **Länge eines Linienzugs** bestimmt sich als Summe der euklidischen Abstände zwischen allen aufeinanderfolgenden Kontrollpunkten. Für einen Linienzug $L_i := (p_1, \ldots, p_n)$ berechnet sich seine Länge demnach durch

$$f_1(L_i) := \sum_{j=1}^{n-1} \delta(p_j, p_{j+1}).$$

Aus den vorgenannten Überlegungen folgt weiterhin, dass Text-Linienzüge aufgrund ihrer häufig geringen Länge oft in verhältnismäßig kurzer Zeit gezeichnet werden können. Demgegenüber weisen Zeichenoperationen für (längere) Form-Linienzüge in der Regel eine höhere Dauer auf.

Aus der zweiten Spalte in Tabelle 3.3 ist ersichtlich, dass die durchschnittliche Dauer der Zeichenoperationen für Text-Linienzüge 309,38 Millisekunden beträgt, während Form-Linienzüge durchschnittlich in 1 372,43 Millisekunden gezeichnet werden. Die maximale gemessene Zeichendauer eines Text-Linienzugs beträgt 985 Millisekunden, während die Erstellung des langsamsten Form-Linienzugs 10 891 Millisekunden in Anspruch nahm.

Die **Dauer der Zeichenoperation** eines Linienzugs $L_i := (p_1, \ldots, p_n)$ bestimmt sich durch die zeitliche Differenz zwischen dem Zeitstempel seines letzten und ersten Kontrollpunkts. Es ergibt sich somit

$$f_2(L_i) := t_{p_n} - t_{p_1}.$$

Weil Text-Linienzüge häufig kürzer sind und in geringerer Zeit gezeichnet werden, kann darüber hinaus davon ausgegangen werden, dass sie in der Regel aus weniger

Kontrollpunkten bestehen als Form-Linienzüge.

Spalte drei in Tabelle 3.3 zeigt, dass Text-Linienzüge durchschnittlich über 14,14 Kontrollpunkte verfügen, während Form-Linienzüge im Durchschnitt aus 108,7 Kontrollpunkten bestehen. Während der längste Form-Linienzug 1 240 Kontrollpunkte aufweist, hat der längste Text-Linienzug lediglich eine Länge von 94 Kontrollpunkten.

Die **Anzahl der Kontrollpunkte** eines Linienzugs L_i ist gegeben durch

$$f_3(L_i) := |L_i|.$$

Erweiterte geometrische Eigenschaften von Linienzügen

Sowohl bei Text-Linienzügen als auch bei Form-Linienzügen sind gelegentliche Selbstüberschneidungen zu beobachten. Bei genauerer Betrachtung stellt man jedoch fest, dass Form-Linienzüge sich häufiger selbst schneiden als Text-Linienzüge.

Der Grund hierfür liegt darin, dass viele Zeichen aus mehreren getrennten Linienzügen bestehen, die sich zwar in der Regel gegenseitig schneiden, allerdings keinen Schnittpunkt mit sich selbst aufweisen. Die Buchstaben „t“ und „f“ werden beispielsweise für gewöhnlich jeweils durch zwei separate Linienzüge erzeugt, indem zunächst ein vertikaler Linienzug gezeichnet wird, der dann von einem horizontalen Linienzug geschnitten wird. Des Weiteren gibt es zahlreiche Zeichen wie beispielsweise „s“, „v“ und „1“, die nur aus einem einzigen Linienzug bestehen, der sich nicht selbst schneidet.

Auch viele Formen wie beispielsweise Kreise und Rauten bestehen nur aus einem einzelnen Linienzug, weil sie von Stakeholdern oft ohne Absetzen des Stiftes in einem durchgängigen Zug gezeichnet werden. Allerdings berührt oder schneidet bei diesen Formen das Ende des Linienzugs häufig dessen Anfang, so dass mindestens eine Selbstüberschneidung vorliegt.

Tabelle 3.4.: Kennzahlen der deskriptiven Statistik für erweiterte geometrische Eigenschaften von Linienzügen.

		Anzahl der Selbstüberschneidungen	Abstand zwischen erstem und letztem Kontrollpunkt in px
Texte	Min	0	0
	Max	5	79,71
	$\overline{x}$	0,15	11,94
	s	0,43	8,3
Formen	Min	0	0
	Max	444	1 752,14
	$\overline{x}$	0,54	54,34
	s	8,44	116,18

Insbesondere bei überdurchschnittlich vielen Schnittpunkten mit sich selbst kann zumeist von einem Form-Linienzug ausgegangen werden. Ein leicht nachzuvollziehendes Beispiel hierfür sind Endknoten in Prozessskizzen: Die ausgefüllten Kreise werden in der Regel ohne Absetzen des Stiftes mit einem einzigen langen Linienzug ausgemalt, der sich unzählige Male selbst schneidet. Bei Zahlen und Buchstaben kommt dieses Phänomen nicht vor.

Die erste Spalte in Tabelle 3.4 zeigt, dass Text-Linienzüge durchschnittlich 0,15 Selbstüberschneidungen aufweisen, während Form-Linienzüge im Durchschnitt 0,54 Schnittpunkte mit sich selbst haben. Für Text-Linienzüge konnten maximal 5 Selbstüberschneidungen gemessen werden, während für Form-Linienzüge maximal 444 Selbstüberschneidungen beobachtet werden konnten.

Um für einen Linienzug L_i die Anzahl seiner Selbstüberschneidungen messen zu können, werden die Verbindungsstrecken zwischen jeweils zwei aufeinanderfolgenden Kontrollpunkten p_j und p_{j+1} betrachtet. Diese werden im Folgenden als *Segmente (des Linienzugs)* bezeichnet. Die Anzahl der Schnittpunkte eines Linienzugs mit sich selbst berechnet sich, indem man die einzelnen Segmente des Linienzugs paarweise auf Überschneidung miteinander testet.

Hierzu sei das Segment zwischen zwei Punkten p_j und p_{j+1} mit $\text{seg}(p_j, p_{j+1})$ be-

zeichnet. Um auszudrücken, dass zwei Segmente $\text{seg}(p_j, p_{j+1})$ und $\text{seg}(p_k, p_{k+1})$ sich schneiden, wird im Folgenden $\text{seg}(p_j, p_{j+1}) \cap \text{seg}(p_k, p_{k+1}) \neq \emptyset$ geschrieben.

Zunächst wird eine Hilfsfunktion $\text{Intersects} : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \{0, 1\}$ definiert, die als Indikator-Funktion für das Vorhandensein eines Schnittpunkts zwischen zwei Segmenten verwendet wird. Es sei

$$\text{Intersects}(p_j, p_{j+1}, p_k, p_{k+1}) := \begin{cases} 1, & \text{falls } \text{seg}(p_j, p_{j+1}) \cap \text{seg}(p_k, p_{k+1}) \neq \emptyset \\ 0, & \text{sonst.} \end{cases}$$

Dann berechnet sich die **Anzahl der Selbstüberschneidungen** eines Linienzugs $L_i := (p_1, \ldots, p_n)$, indem alle Segmente des Linienzugs der Reihe nach durch Anwendung der zuvor definierten Indikator-Funktion auf Überschneidung mit jeweils allen nachfolgenden Segmenten getestet werden. Die entsprechende Formel lautet:

$$f_4(L_i) := \sum_{j=1}^{n-3} \sum_{k=j+2}^{n-1} \text{Intersects}(p_j, p_{j+1}, p_k, p_{k+1}).$$

Darüber hinaus lässt sich beobachten, dass Formen sich im Vergleich zu einzelnen Zeichen häufig verhältnismäßig weit in horizontaler und/oder vertikaler Richtung ausdehnen. Beispielsweise sind Pfeile in einem Process Canvas oder Assoziationsbeziehungen in einem Object Canvas im Vergleich zu Texten wesentlich breiter und/oder höher. Entsprechend groß ist der Abstand zwischen dem ersten und letzten Kontrollpunkt solcher Form-Linienzüge.

Demgegenüber ist der Abstand zwischen Start- und Endpunkt bei Text-Linienzügen zumeist deutlich geringer. Dies ist schon allein deshalb der Fall, weil Text-Linienzüge wie zuvor erläutert in der Regel wesentlich kürzer sind als Form-Linienzüge.

Spalte 2 in Tabelle 3.4 zeigt, dass der durchschnittliche euklidische Abstand zwischen dem ersten und letzten Kontrollpunkt von Text-Linienzügen 11,94 Pixel beträgt, während er für Form-Linienzüge bei 54,34 Pixeln liegt. Der längste gemessene Abstand zwischen Start- und Endpunkt eines Text-Linienzugs liegt bei 79,71 Pixeln, während er für Form-Linienzüge 1 752,14 Pixel beträgt.

Tabelle 3.5.: Kennzahlen der deskriptiven Statistik für grundlegende geometrische Eigenschaften der Bounding Boxen von Linienzügen.

		Breite der Bounding Box in px	Höhe der Bounding Box in px	Flächeninhalt der Bounding Box in px^2
Texte	Min	0	0	0
	Max	49,43	73,99	3 092,2
	$\overline{x}$	7,48	12,11	108,43
	s	4,65	9,12	138,97
Formen	Min	0	0	0
	Max	1 804,36	836,51	497 884,06
	$\overline{x}$	94,85	50,33	8 482,07
	s	123,63	66,34	23 968,01

Der **euklidische Abstand zwischen dem ersten und letzten Kontrollpunkt** eines Linienzugs $L_i := (p_1, \ldots, p_n)$ bestimmt sich durch

$$f_5(L_i) := \delta(p_1, p_n).$$

Grundlegende Eigenschaften von Bounding Boxen

Aus der Geometrie der Bounding Box eines Linienzugs können weitere nützliche Informationen abgeleitet werden. Wie zuvor gezeigt, bestehen Zeichen oft aus kürzeren Linienzügen, während Formen eher aus längeren Linienzügen zusammengesetzt sind. Hieraus folgt unmittelbar, dass Bounding Boxen von Form-Linienzügen häufig breiter sind als Bounding Boxen von Text-Linienzügen.

Spalte 1 in Tabelle 3.5 zeigt, dass die durchschnittliche Breite der Bounding Boxen von Text-Linienzügen 7,48 Pixel beträgt, während Bounding Boxen von Form-Linienzügen im Durchschnitt 94,85 Pixel breit sind. Die breiteste beobachtete Bounding Box eines Text-Linienzugs hat eine Breite von 49,43 Pixeln, während die breiteste Bounding Box eines Form-Linienzugs eine Breite von 1 804,36 Pixeln aufweist.

Die **Breite der Bounding Box** eines Linienzugs L_i bestimmt sich als Differenz zwischen dem minimalen und maximalen x-Wert aller Kontrollpunkte. Es ist somit

$$f_6(L_i) := \mathcal{B}(L_i)_{x_{max}} - \mathcal{B}(L_i)_{x_{min}}.$$

Die zuvor getroffenen Aussagen zur Breite von Bounding Boxen lassen sich im Wesentlichen auf deren Höhe übertragen: Bounding Boxen von Form-Linienzügen sind häufig höher als Bounding Boxen von Text-Linienzügen.

Dies ist aus Spalte 2 in Tabelle 3.5 ersichtlich: Während die durchschnittliche Höhe der Bounding Boxen von Text-Linienzügen 12,11 Pixel beträgt, sind Bounding Boxen von Form-Linienzügen im Durchschnitt 50,33 Pixel hoch. Die maximal gemessene Höhe einer Bounding Box eines Text-Linienzugs beträgt 73,99 Pixel, während die höchste Bounding Box eines Form-Linienzugs eine Höhe von 836,51 Pixeln aufweist.

Die **Höhe der Bounding Box** eines Linienzugs L_i bestimmt sich analog zu ihrer Breite als Differenz zwischen dem minimalen und maximalen y-Wert aller Kontrollpunkte, also als Abstand zwischen dem tiefsten und höchsten Punkt des Linienzugs. Es ist demnach

$$f_7(L_i) := \mathcal{B}(L_i)_{y_{max}} - \mathcal{B}(L_i)_{y_{min}}.$$

Aus den vorgenannten Beobachtungen lässt sich weiterhin folgern, dass der Flächeninhalt der Bounding Boxen von Form-Linienzügen in der Regel wesentlich größer ist als der Flächeninhalt der Bounding Boxen von Text-Linienzügen.

Spalte 3 in Tabelle 3.5 zeigt, dass die durchschnittliche Fläche der Bounding Boxen von Text-Linienzügen 108,43 px^2 beträgt, während Bounding Boxen von Form-Linienzügen im Durchschnitt einen Flächeninhalt von 8 482,07 px^2 aufweisen. Die größte Bounding Box eines Text-Linienzugs hat eine Fläche von 3 092,2 px^2, während die größte Bounding Box eines Form-Linienzugs über einen Flächeninhalt von 497 884,06 px^2 verfügt.

Weil Bounding Boxen per Definition Rechtecke in der euklidischen Ebene sind, bestimmt sich der **Flächeninhalt der Bounding Box** eines Linienzugs L_i als Produkt aus Breite und Höhe. Es ist somit

$$f_8(L_i) := f_6(L_i) \cdot f_7(L_i).$$

Es ist auffällig, dass die analysierten Skizzen Bounding Boxen enthalten, die eine Höhe und/oder Breite von 0 Pixeln (und somit einen Flächeninhalt von 0 px^2) aufweisen. Hierbei handelt es sich zum einen um Bounding Boxen von entarteten Text-Linienzügen, die nur aus einem einzigen Kontrollpunkt bestehen. Zum anderen handelt es sich um kurze Form-Linienzüge, die exakt horizontal oder senkrecht verlaufen. Solche Linienzüge weisen keine messbare Höhe oder Breite auf.

Erweiterte Eigenschaften von Bounding Boxen

Wie zuvor diskutiert sind Bounding Boxen von Form-Linienzügen in der Regel breiter und/oder höher als Bounding Boxen von Text-Linienzügen. Dies spiegelt sich unmittelbar im diagonalen Durchmesser der jeweiligen Bounding Box wieder, welcher entsprechend für Form-Linienzüge häufig größer ist als für Text-Linienzüge.

Spalte 4 in Tabelle 3.6 zeigt, dass der durchschnittliche diagonale Durchmesser der Bounding Boxen von Text-Linienzügen 14,93 Pixel beträgt, während er sich für die Bounding Boxen von Form-Linienzügen auf 115,44 Pixel beläuft. Der höchste gemessene diagonale Durchmesser einer Bounding Box eines Text-Linienzugs beträgt 84,98 Pixel, während der maximale Wert für Form-Linienzüge bei 1 813,72 Pixeln liegt.

Der **diagonale Durchmesser der Bounding Box** eines Linienzugs L_i bestimmt sich als euklidischer Abstand zwischen ihrem unteren linken und oberen rechten Eckpunkt gemäß

$$f_9(L_i) := \delta(\mathcal{B}(L_i)_{min}, \mathcal{B}(L_i)_{max}).$$

Tabelle 3.6.: Kennzahlen der deskriptiven Statistik für erweiterte Eigenschaften der Bounding Boxen von Linienzügen.

		Diagonaler Durchmesser der Bounding Box in px	Winkel unter der Diagonale der Bounding Box in rad
Texte	Min	0	0
	Max	84,98	1,56
	$\overline{x}$	14,93	0,9
	s	9,18	0,39
Formen	Min	0	0
	Max	1 813,72	1,56
	$\overline{x}$	115,44	0,62
	s	133,74	0,44

Weiterhin überlegt man sich, dass der Winkel zwischen der unteren Kante einer Bounding Box und ihrer Diagonalen von ihrem Seitenverhältnis abhängt: Je höher eine Bounding Box im Verhältnis zu ihrer Breite ist, desto größer ist der betrachtete Winkel. Für flache Bounding Boxen ist der entsprechende Winkel somit eher klein, während er für hohe Bounding Boxen eher groß ist.

Die meisten Zeichen besitzen entweder eine annähernd quadratische Bounding Box, was auf viele Kleinbuchstaben wie beispielsweise „a“, „e“ und „n“ zutrifft, oder ihre Höhe übertrifft ihre Breite, was insbesondere, aber nicht ausschließlich, für Großbuchstaben und Zahlen gilt.

Bei Formen kann häufig das Gegenteil beobachtet werden: Viele Skizzen werden von links nach rechts gezeichnet, weshalb viele Formen wie beispielsweise Pfeile und Rechtecke ebenfalls häufig horizontal verlaufen. Aus diesem Grund ist ihre Breite oft wesentlich größer als ihre Höhe.

Hieraus folgt, dass Bounding Boxen von Text-Linienzügen häufig einen Winkel $\geq 45\,^\circ$ unter ihrer Diagonalen aufweisen, weil ihre Höhe oft größer-gleich ihrer Breite ist. Weil Form-Linienzüge demgegenüber oftmals breiter als höher sind, weisen deren Bounding-Box-Diagonalen entsprechend häufig einen Winkel $< 45\,^\circ$ auf.

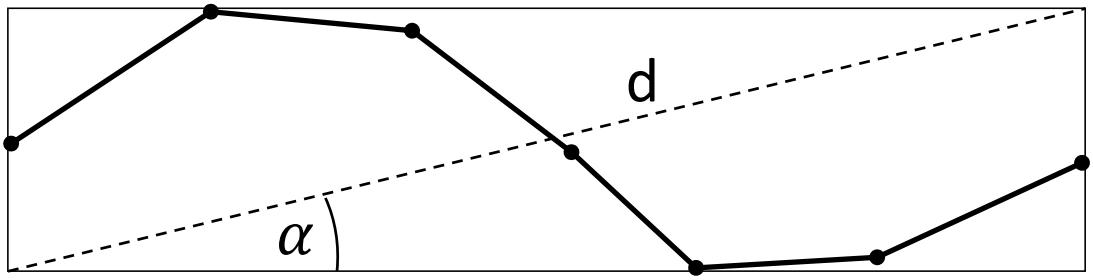

Abbildung 3.5.: Bounding Box eines Linienzugs mit eingezeichnetem Winkel α unterhalb der Diagonale d. Die Diagonale bildet zusammen mit der unteren und rechten Kante der Bounding Box ein rechtwinkliges Dreieck.

Spalte 2 in Tabelle 3.6 zeigt die statistischen Kennzahlen für die Winkel der Bounding-Box-Diagonalen von Text- und Form-Linienzügen als Werte im Bogenmaß. Während für Bounding Boxen von Text-Linienzügen durchschnittlich ein Winkel von 0,9 rad ($\approx$ 51,57 °) zwischen ihrer Diagonalen und der unteren Kante der Bounding Box beobachtet werden kann, weisen Bounding Boxen von Form-Linienzügen im Durchschnitt einen geringeren Winkel von 0,62 rad ($\approx$ 35,52 °) auf.

Zur Berechnung des Winkels unterhalb der Bounding-Box-Diagonalen macht man sich klar, dass die Diagonale einer Bounding Box zusammen mit ihrer unteren und rechten Kante ein rechtwinkliges Dreieck bildet. Dies ist in Abbildung 3.5 illustriert. Deshalb gilt für den Winkel α unterhalb der Diagonale $\tan\alpha = \frac{Gegenkathete}{Ankathete}$ [202].

Für einen Linienzug L_i ist die Gegenkathete des Winkels α durch die Höhe $f_7(L_i)$ seiner Bounding Box gegeben, während die Ankathete durch die Breite $f_6(L_i)$ gegeben ist. Somit ergibt sich $\tan\alpha = \frac{f_7(L_i)}{f_6(L_i)}$.

Zur Bestimmung des Winkels α kann nun die Arcus-Tangens-Funktion verwendet werden [202].

Damit berechnet sich der **Winkel unter der Bounding-Box-Diagonalen** für

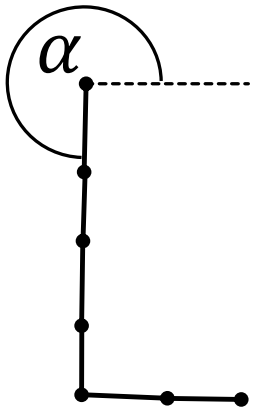

Abbildung 3.6.: Initialer Winkel α eines Linienzugs für den Buchstaben „L“.

einen Linienzug L_i durch

$$f_{10}(L_i) := \alpha = \arctan(\tan\alpha) = \arctan(\frac{f_7(L_i)}{f_6(L_i)}).$$

Krümmungsverhalten von Linienzügen

Es lässt sich beobachten, dass der initiale Linienzug für viele Zeichen häufig in einem steilen Winkel von oben nach unten geführt wird. So starten beispielsweise die Linienzüge für die Buchstaben „L“, „m“, „P“ und „r“ jeweils an der linken oberen Spitze und verlaufen dann zunächst senkrecht nach unten. Für diese Text-Linienzüge ergibt sich somit ein steiler initialer Winkel nahe 270 °. Abbildung 3.6 illustriert diese Eigenschaft für den initialen Winkel des Buchstabens „L“.

Demgegenüber verlaufen viele Formen wie zuvor erläutert häufig eher horizontal von links nach rechts in einem flacheren Winkel. Für diese Form-Linienzüge ist somit ein initialer Winkel unter 90 ° oder über 270 ° zu erwarten.

Gemäß eines Vorschlags von Rubine [196] wird für diese Eigenschaft im Folgenden nicht der Winkel selbst sondern der Kosinus des Winkels betrachtet, um eine Diskontinuität zu vermeiden, die bei Überschreitung eines Winkels von 360 ° auftreten

Tabelle 3.7.: Kennzahlen der deskriptiven Statistik für das Krümmungsverhalten von Linienzügen.

		Kosinus des initialen Winkels	Kosinus des Winkels zwischen erstem und letztem Kontrollpunkt
Texte	Min	−1	−1
	Max	1	1
	$\overline{x}$	0,01	0,35
	s	0,65	0,52
Formen	Min	−1	−1
	Max	1	1
	$\overline{x}$	0,29	0,28
	s	0,6	0,66

würde.

Dementsprechend ist für den initialen Winkel von Text-Linienzügen ein Kosinus-Wert nahe Null zu erwarten (vgl. Winkeltabelle A.1 sowie zugehörige Abbildung A.1 im Anhang), während für Form-Linienzüge häufig ein höherer Kosinus-Wert angenommen werden kann.

Spalte 1 in Tabelle 3.7 zeigt Kennzahlen der deskriptiven Statistik für den Kosinus des initialen Winkels für Text- und Form-Linienzüge. Man sieht, dass der durchschnittliche Kosinus des initialen Winkels für Text-Linienzüge mit 0,01 nahe bei Null liegt, was einem Winkel von etwa 90 bzw. 270 Grad entspricht. Für Form-Linienzüge ist der durchschnittliche Kosinus des initialen Winkels größer und entspricht mit 0,29 einem flacheren Winkel nahe 70 bzw. 290 Grad.

In Abschnitt 3.2.8 wurde bereits gezeigt, dass der Kosinus des eingeschlossenen Winkels α zwischen zwei Vektoren $q, d \in \mathbb{R}^n$ mit Hilfe der folgenden geometrischen Formel bestimmt werden kann:

$$\cos(\alpha) = \frac{q \cdot d}{|q| \cdot |d|}.$$

Für einen Linienzug $L_i := (p_1, \ldots, p_n)$ lassen sich die Vektoren $q, d \in \mathbb{R}^2$ wie folgt definieren:

Der Vektor q sei der Verbindungsvektor von p_1 zu p_2, also

$$q := p_2 - p_1 = (x_{p_2} - x_{p_1}, y_{p_2} - y_{p_1}).$$

Als Vektor d kann der Einheitsvektor $(1, 0)$ gewählt werden, also

$$d := (1, 0).$$

Dann sind $q \cdot d = x_{p_2} - x_{p_1}$ und $|q| \cdot |d| = \sqrt{(x_{p_2} - x_{p_1})^2 + (y_{p_2} - y_{p_1})^2} = \delta(p_1, p_2)$.

Somit berechnet sich der **Kosinus des initialen Winkels** für einen Linienzug $L_i := (p_1, \ldots, p_n)$ durch

$$f_{11}(L_i) := \frac{x_{p_2} - x_{p_1}}{\delta(p_1, p_2)}.$$

Weiterhin lässt sich feststellen, dass die meisten Stakeholder nicht nur von oben nach unten sondern auch von links nach rechts schreiben. Hieraus folgt, dass sich der Endpunkt von Text-Linienzügen häufig rechts-unterhalb des Startpunkts befindet. Somit ergibt sich für Text-Linienzüge oft ein Winkel zwischen 270 und 315 Grad zwischen seinem ersten und letzten Kontrollpunkt. Dies gilt für viele Buchstaben wie beispielsweise „R“, „M“ und „Z“ sowie auch für einige Zahlen wie beispielsweise „1“ und „2“. Abbildung 3.7 illustriert diesen Sachverhalt für den Buchstaben „L“.

Spalte 2 in Tabelle 3.7 zeigt, dass der durchschnittliche Kosinus-Wert des Winkels zwischen erstem und letztem Kontrollpunkt für Text-Linienzüge bei 0,35 liegt. Dies entspricht einem Winkel von etwa 70 bzw. 290 Grad. Demgegenüber liegt der entsprechende Wert für Form-Linienzüge bei 0,28, was einem steileren Winkel von etwa 74 bzw. 286 Grad entspricht.

Text-Linienzüge weisen somit im Durchschnitt einen flacheren Winkel zwischen erstem und letztem Kontrollpunkt auf als Form-Linienzüge, deren Winkel häufig näher an 90 bzw. 270 Grad liegt. Insbesondere lassen sich durch Betrachtung dieses Merkmals Zeichen von solchen Formen unterscheiden, die in einem sehr steilen Winkel und/oder von rechts nach links gezeichnet werden.

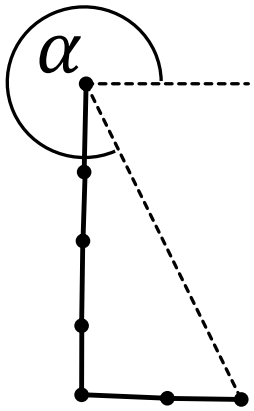

Abbildung 3.7.: Winkel α zwischen erstem und letztem Kontrollpunkt eines Linienzugs für den Buchstaben „L“.

Zur Berechnung wird die gleiche Formel verwendet, die bereits zur Bestimmung des initialen Winkels genutzt wurde. Der wesentliche Unterschied besteht in der Wahl des Vektors q. Während d erneut als Einheitsvektor $d := (1, 0)$ gesetzt wird, ist q nun der Verbindungsvektor von p_1 zu p_n, also

$$q := p_n - p_1 = (x_{p_n} - x_{p_1}, y_{p_n} - y_{p_1}).$$

Damit ist $q \cdot d = x_{p_n} - x_{p_1}$ und $|q| \cdot |d| = \sqrt{(x_{p_n} - x_{p_1})^2 + (y_{p_n} - y_{p_1})^2} = \delta(p_1, p_n)$.

Der **Kosinus des Winkels zwischen erstem und letztem Kontrollpunkt** eines Linienzugs L_i berechnet sich somit durch

$$f_{12}(L_i) := \frac{x_{p_n} - x_{p_1}}{\delta(p_1, p_n)}.$$

3.4.5. Globale Features zur Klassifizierung von Linienzügen

Zusätzlich zu den bisher vorgestellten lokalen Features konnten außerdem 12 globale Features zur Klassifizierung von Linienzügen identifiziert werden. Im Unterschied zu den lokalen Eigenschaften berücksichtigen diese insbesondere die Beziehung eines

betrachteten Linienzugs zu seinen direkten Vorgängern und Nachfolgern sowie zu seinen temporalen und räumlichen Nachbarn. Für einen Linienzug L_i bilden sie die Werte $f_{13}(L_i), \ldots, f_{24}(L_i)$ des charakterisierenden Vektors $f(L_i)$.

Hierbei ist zu beachten, dass die zeitlichen Abstände zwischen Linienzügen in den digital rekonstruierten Skizzen von den realen zeitlichen Distanzen, die bei der Erstellung der analogen Originalskizzen auftraten, abweichen können. Es kann angenommen werden, dass die zeitlichen Abstände zwischen bestimmten Linienzügen in einem realen Workshop größer sind als bei der Rekonstruktion, weil Stakeholder in einem Workshop gelegentliche Denk- und Diskussionspausen einlegen müssen, die bei der Rekonstruktion entfallen.

Jedoch lässt sich argumentieren, dass mindestens solche Linienzüge, die zusammenhängende Elemente bilden, auch in einer echten Workshop-Situation für gewöhnlich in schneller Abfolge direkt hintereinander gezeichnet werden. Dies trifft insbesondere auf Texte zu, weil Stakeholder normalerweise nicht während der Erstellung eines Wortes pausieren. Somit lässt sich annehmen, dass zumindest einzelne Buchstaben zusammenhängender Texte ohne Pause direkt hintereinander geschrieben werden und die zeitlichen Abstände zwischen den entsprechenden Linienzügen in den Rekonstruktionen deshalb denen in den Originalskizzen ähneln. Weil 89,72 % aller Linienzüge Text-Linienzüge sind, dürften somit die meisten zeitlichen Abstände zwischen Linienzügen in den rekonstruierten Skizzen repräsentativ sein.

Darüber hinaus wird sicherlich auch die Reihenfolge der Linienzüge in den digitalen Rekonstruktionen nicht immer mit der Reihenfolge übereinstimmen, in der diese im Workshop gezeichnet wurden. Dies trifft insbesondere dann zu, wenn Elemente, die zu einem früheren Zeitpunkt erstellt wurden, radiert oder korrigiert wurden.

Allerdings kann auch hier argumentiert werden, dass Texte stets von links nach rechts geschrieben werden. Die Schreibrichtung entspricht somit sowohl in den Rekonstruktionen als auch in den originalen analogen Skizzen der Leserichtung. Es lässt sich somit annehmen, dass mindestens die Text-Linienzüge, die zusammenhängende Wörter und Sätze bilden, in den rekonstruierten Skizzen in der gleichen Reihenfolge von links nach rechts geschrieben wurden wie in den Originalzeichnungen. Weil die meisten Linienzüge wie zuvor beschrieben Text-Linienzüge sind, dürfte folglich auch

die Reihenfolge der Linienzüge in den rekonstruierten Skizzen in den meisten Fällen mit der ursprünglichen Reihenfolge übereinstimmen.

Obwohl die digitalen Rekonstruktionen somit wahrscheinlich Linienzüge enthalten, deren Reihenfolgen und zeitliche Abstände von den originalen Skizzen abweichen, verfälscht dies nicht die gesamte Analyse einer Skizze. Es wirkt sich in der Regel nur auf die direkten Vorgänger und Nachfolger eines betroffenen Linienzugs sowie auf seine unmittelbaren zeitlichen und räumlichen Nachbarn aus und beeinflusst deshalb jeweils nur seine direkte Umgebung.

Die globalen Features zur Klassifizierung von Linienzügen werden im Folgenden hergeleitet und mathematisch präzise definiert. Überdies werden die in Abschnitt 3.2.9 vorgestellten Kennzahlen der deskriptiven Statistik für jedes Merkmal untersucht.

Grundlegende Nachbarschaftsbeziehungen zwischen Linienzügen

Zunächst überlegt man sich, dass die meisten Texte in Interaction-Room-Canvases innerhalb einer Umrandung stehen. Dies trifft beispielsweise auf Beschriftungen von Aktionen in Process Canvases, auf Namen von Objekten in Object Canvases sowie auf Features in Feature Canvases zu. Für die meisten Text-Linienzüge, aus denen die entsprechenden Texte zusammengesetzt sind, ist somit zu erwarten, dass sie vollständig in den Bounding Boxen ihrer umgebenden Form-Linienzüge enthalten sind.

Demgegenüber sind Form-Linienzüge jedoch nur selten vollständig innerhalb der Bounding Box eines anderes Linienzugs zu finden. Die meisten Formen, die häufig in Interaction-Room-Skizzen verwendet werden, wie beispielsweise Rechtecke, Rauten, Pfeile und Kreise, stehen fast niemals innerhalb umgebender Umrandungen.

Die erste Spalte in Tabelle 3.8 zeigt, dass Text-Linienzüge durchschnittlich in 0,94 Bounding Boxen anderer Linienzüge enthalten sind. Somit wird folglich fast jeder Text-Linienzug zusätzlich zu seiner eigenen Bounding Box ebenfalls von der Bounding Box eines umgebenden Linienzugs umfasst. Demgegenüber existieren durchschnittlich jedoch nur 0,16 umfassende Bounding Boxen für Form-Linienzüge. Form-

Tabelle 3.8.: Kennzahlen der deskriptiven Statistik für grundlegende Nachbarschaftsbeziehungen zwischen Linienzügen.

		Anzahl umfassender Bounding Boxen	Anzahl vollständig enthaltener Linienzüge
Texte	Min	0	0
	Max	4	2
	$\overline{x}$	0,94	0,02
	s	0,54	0,15
Formen	Min	0	0
	Max	4	344
	$\overline{x}$	0,16	7,94
	s	0,41	16,59

Linienzüge werden somit im Durchschnitt nur selten von zusätzlichen Bounding Boxen umgeben, während dies für Text-Linienzüge fast immer der Fall ist.

Zur Bestimmung der Anzahl der Bounding Boxen, die einen betrachteten Linienzug umfassen, überlegt man sich zunächst, dass eine Bounding Box $\mathcal{B}(L_i)$ einen Linienzug L_j genau dann vollständig enthält, wenn $\mathcal{B}(L_j)$ vollständig in $\mathcal{B}(L_i)$ enthalten ist. Dies kann durch eine Hilfsfunktion

$$\text{Contains} : \mathcal{B} \times \mathcal{B} \to \{0, 1\}$$

beschrieben werden mit

$$\text{Contains}(\mathcal{B}(L_i), \mathcal{B}(L_j)) := \begin{cases} 1, & \text{falls } \mathcal{B}(L_i)_{x_{min}} \leq \mathcal{B}(L_j)_{x_{min}} \\ & \wedge\ \mathcal{B}(L_i)_{x_{max}} \geq \mathcal{B}(L_j)_{x_{max}} \\ & \wedge\ \mathcal{B}(L_i)_{y_{min}} \leq \mathcal{B}(L_j)_{y_{min}} \\ & \wedge\ \mathcal{B}(L_i)_{y_{max}} \geq \mathcal{B}(L_j)_{y_{max}} \\ 0, & \text{sonst.} \end{cases}$$

Wenn alle vier Eckpunkte einer Bounding Box $\mathcal{B}(L_j)$ in einer Bounding Box $\mathcal{B}(L_i)$ enthalten sind, so ist die gesamte Bounding Box $\mathcal{B}(L_j)$ in $\mathcal{B}(L_i)$ enthalten und damit

folglich auch der komplette Linienzug L_j. In diesem Fall liefert die Funktion Contains einen Wert von 1 zurück, andernfalls 0.

Die **Anzahl der umfassenden Bounding Boxen** für einen Linienzug $L_i \in L(S)$ lässt sich somit bestimmen durch

$$f_{13}(L_i) := \sum_{\substack{L_j \in L(S) \\ L_j \neq L_i}} \text{Contains}(\mathcal{B}(L_j), \mathcal{B}(L_i)).$$

Aus den vorangegangenen Überlegungen folgt unmittelbar, dass Bounding Boxen von Form-Linienzügen häufig mehrere Text-Linienzüge umfassen. Umgekehrt ist dies in der Regel jedoch nicht der Fall: Bounding Boxen von Text-Linienzügen enthalten für gewöhnlich keine anderen Linienzüge vollständig.

Spalte 2 in Tabelle 3.8 zeigt, dass jeder Text-Linienzug durchschnittlich nur 0,02 andere Linienzüge vollständig mit seiner Bounding Box umschließt. Demgegenüber umfasst jedoch jede Bounding Box eines Form-Linienzugs durchschnittlich 7,94 andere Linienzüge. Es existiert sogar ein Form-Linienzug, dessen Bounding Box 344 andere Linienzüge vollständig enthält. Hierbei handelt es sich um die Umrandung einer Aktion auf einem Process Canvas, die einen verhältnismäßig langen mehrzeiligen Text einfasst.

Es lässt sich somit folgern, dass es sich bei Linienzügen, deren Bounding Boxen mehrere andere Linienzüge vollständig umschließen, mit hoher Wahrscheinlichkeit um Form-Linienzüge handelt.

Die **Anzahl der vollständig enthaltenen Linienzüge**, d. h. die Anzahl der Linienzüge, die vollständig in der Bounding Box eines Linienzugs $L_i \in L(S)$ enthalten sind, lässt sich unter Anwendung der zuvor definierten Contains-Funktion bestimmen durch

$$f_{14}(L_i) := \sum_{\substack{L_j \in L(S) \\ L_j \neq L_i}} \text{Contains}(\mathcal{B}(L_i), \mathcal{B}(L_j)).$$

Der wesentliche Unterschied zwischen den Formeln f_{13} und f_{14} besteht darin, dass die Argumente für die Contains-Funktion permutiert sind.

Tabelle 3.9.: Kennzahlen der deskriptiven Statistik für zeitliche Distanzen zwischen Linienzügen.

		Zeitlicher Abstand zum vorherigen Linienzug in ms	Zeitlicher Abstand zum nachfolgenden Linienzug in ms
Texte	Min	0	0
	Max	4 075 703	125 080
	$\overline{x}$	658,68	319,08
	s	19 920,48	1 583,93
Formen	Min	0	0
	Max	1 090 468	4 075 703
	$\overline{x}$	2 188,67	5 181,19
	s	20 862,86	62 119,24

Zeitliche Distanzen zwischen Linienzügen

Als nächstes werden die zeitlichen Abstände von Linienzügen zu ihren unmittelbaren Vorgängern und Nachfolgern betrachtet. Es wurde bereits diskutiert, dass einzelne Zeichen eines Wortes in der Regel in rascher Abfolge direkt hintereinander geschrieben werden. Es ist deshalb zwischen den betreffenden Text-Linienzügen oft ein geringer zeitlicher Abstand zu erwarten. Demgegenüber kann für Form-Linienzüge häufig ein größerer zeitlicher Abstand zum vorherigen und nachfolgenden Linienzug festgestellt werden.

Aus der ersten Spalte in Tabelle 3.9 ist ersichtlich, dass der zeitliche Abstand eines Text-Linienzugs zu seinem unmittelbaren Vorgänger im Durchschnitt 658,68 ms beträgt. Für Form-Linienzüge ist dieser Abstand mit durchschnittlich 2 188,67 ms wesentlich größer. Als maximale zeitliche Distanz zum vorherigen Linienzug konnten sowohl für Text-Linienzüge als auch für Form-Linienzüge sehr große Werte beobachtet werden. Dies lässt darauf schließen, dass die Stakeholder während der Anfertigung der entsprechenden Skizze(n) eine längere Pause gemacht haben.

Der **zeitliche Abstand zum vorherigen Linienzug** kann für einen Linienzug

$L_i \in L(S)$ bestimmt werden durch

$$f_{15}(L_i) := \begin{cases} \tau(L_i, L_{i-1}), & \text{falls } i > 1 \\ 0, & \text{sonst.} \end{cases}$$

Für den zeitlichen Abstand eines Linienzugs zu seinem Nachfolger gilt im Wesentlichen das Gleiche wie für den zeitlichen Abstand zu seinem Vorgänger. Auch hier ist zu erwarten, dass die zeitliche Differenz zwischen Text-Linienzügen und dem unmittelbar nachfolgenden Linienzug oft geringer ist als zwischen Form-Linienzügen und ihrem direkten Nachfolger.

Spalte 2 in Tabelle 3.9 zeigt, dass Text-Linienzüge durchschnittlich einen zeitlichen Abstand von 319,08 ms zum nachfolgenden Linienzug aufweisen. Demgegenüber haben Form-Linienzüge im Durchschnitt eine zeitliche Distanz von 5 181,19 ms zu ihrem Nachfolger.

Der **zeitliche Abstand zum nachfolgenden Linienzug** bestimmt sich für einen Linienzug $L_i \in L(S)$ durch

$$f_{16}(L_i) := \begin{cases} \tau(L_i, L_{i+1}), & \text{falls } i < |L(S)| \\ 0, & \text{sonst.} \end{cases}$$

Euklidischer Abstand zwischen Linienzügen

Zeitlich aufeinanderfolgende Zeichen eines Textes weisen häufig einen geringen euklidischen Abstand zueinander auf, sofern sie nicht durch Leerzeichen oder einen Zeilenumbruch voneinander getrennt sind. Wenn ein Zeichen aus mehreren Text-Linienzügen besteht, so liegt zwischen diesen häufig sogar eine Überschneidung vor.

Demgegenüber lässt sich beobachten, dass Form-Linienzüge zu vorausgegangenen und nachfolgenden Linienzügen oft eine größere euklidische Distanz aufweisen, als dies für Text-Linienzüge der Fall ist.

Tabelle 3.10.: Kennzahlen der deskriptiven Statistik für euklidische Distanzen zwischen Linienzügen.

		Euklidischer Abstand zum vorherigen Linienzug in px	Euklidischer Abstand zum nachfolgenden Linienzug in px
Texte	Min	0	0
	Max	1 752,02	6 441,22
	$\overline{x}$	25,69	24,02
	s	64,25	56,74
Formen	Min	0	0
	Max	6 441,22	1 979,22
	$\overline{x}$	117,59	132,79
	s	168,05	183,28

Spalte 1 in Tabelle 3.10 zeigt, dass Text-Linienzüge eine durchschnittliche euklidische Distanz von 25,69 Pixeln zum vorhergehenden Linienzug aufweisen. Für Form-Linienzüge liegt ein wesentlich höherer durchschnittlicher Abstand von 117,59 Pixeln vor.

Der **euklidische Abstand zum vorherigen Linienzug** kann für einen Linienzug $L_i \in L(S)$ berechnet werden durch

$$f_{17}(L_i) := \begin{cases} \Delta(L_i, L_{i-1}), & \text{falls } i > 1 \\ 0, & \text{sonst.} \end{cases}$$

Für die Distanz eines Linienzugs zu seinem unmittelbaren Nachfolger gilt im Wesentlichen das gleiche wie für den Abstand zu seinem Vorgänger. Auch hier ist für Text-Linienzüge durchschnittlich ein wesentlich kleinerer Wert zu beobachten als für Form-Linienzüge.

Aus Spalte 2 in Tabelle 3.10 ist ersichtlich, dass Text-Linienzüge im Durchschnitt eine euklidische Distanz von 24,02 Pixeln zum nachfolgenden Linienzug aufweisen. Für Form-Linienzüge ist der durchschnittliche Abstand mit 132,79 Pixeln wesentlich größer.

Der **euklidische Abstand zum nachfolgenden Linienzug** berechnet sich für einen Linienzug $L_i \in L(S)$ durch folgende Formel:

$$f_{18}(L_i) := \begin{cases} \Delta(L_i, L_{i+1}), & \text{falls } i < |L(S)| \\ 0, & \text{sonst.} \end{cases}$$

Temporale Nachbarn

Als nächstes werden solche Linienzüge betrachtet, die zu einem gegebenen Linienzug eine spezifizierte maximale zeitliche Distanz aufweisen. Diese Linienzüge werden im Folgenden als *temporale Nachbarn* bezeichnet. Verschiedene Forschergruppen schlagen vor, Linienzüge als temporale Nachbarn zu betrachten, wenn zwischen ihnen eine zeitliche Differenz von maximal 3,5 Sekunden besteht [59, 238]. Dementsprechend sei für einen Linienzug $L_i \in L(S)$ die Menge seiner temporalen Nachbarn gegeben durch

$$T(L_i) := \{L_j \in L(S) \mid L_j \neq L_i \wedge \tau(L_i, L_j) \leq 3\,500\}.$$

Die Menge $T(L_i)$ enthält somit alle Linienzüge einer betrachteten Skizze, deren zeitlicher Abstand zu L_i 3,5 Sekunden (= 3 500 Millisekunden) oder weniger beträgt.

Wie zuvor diskutiert werden einzelne Zeichen eines Wortes in der Regel in rascher Abfolge ohne Unterbrechung direkt hintereinander geschrieben. Zudem ist die durchschnittliche Dauer der Zeichenoperation für Text-Linienzüge wesentlich geringer als für Form-Linienzüge. Hieraus folgt die Annahme, dass die Anzahl temporaler Nachbarn für Text-Linienzüge höher ist als für Form-Linienzüge.

Spalte 1 in Tabelle 3.11 bestätigt dies: Man sieht, dass Text-Linienzüge durchschnittlich über 12,19 temporale Nachbarn verfügen. Demgegenüber hat jeder Form-Linienzug im Durchschnitt nur 4,82 temporale Nachbarn.

Die **Anzahl temporaler Nachbarn** eines Linienzugs L_i entspricht der Anzahl der Elemente in der zuvor eingeführten Menge $T(L_i)$. Es ist somit

$$f_{19}(L_i) := |T(L_i)|.$$

Tabelle 3.11.: Kennzahlen der deskriptiven Statistik für temporale Nachbarn von Linienzügen.

		Anzahl temporaler Nachbarn	Durchschnittliche euklidische Distanz zu temporalen Nachbarn in px	Durchschnittliche Länge der temporalen Nachbarn in px
Texte	Min	0	0	0
	Max	23	264,8	585,9
	$\overline{x}$	12,19	31,54	35,44
	s	3,35	12,95	24,07
Formen	Min	0	0	0
	Max	19	559,81	2 537,44
	$\overline{x}$	4,82	38,6	103,88
	s	3,19	53,01	148,45

Weiterhin kann beobachtet werden, dass die durchschnittliche euklidische Distanz von Text-Linienzügen zu ihren temporalen Nachbarn im Durchschnitt geringer ist als der Abstand von Form-Linienzügen zu ihren temporalen Nachbarn. Dies folgt aus der bereits zuvor geschilderten Beobachtung, dass Text-Linienzüge nicht nur in rascher Abfolge schnell hintereinander gezeichnet werden sondern in der Regel auch dicht beisammen stehen.

Die zweite Spalte in Tabelle 3.11 zeigt, dass der durchschnittliche Abstand zu temporalen Nachbarn für Text-Linienzüge 31,54 Pixel beträgt, während er sich für Form-Linienzüge auf 38,6 Pixel beläuft.

Der **durchschnittliche euklidische Abstand zu temporalen Nachbarn** bestimmt sich als Quotient aus der Summe der Abstände des betrachteten Linienzugs L_i zu seinen temporalen Nachbarn und der Anzahl dieser. Die entsprechende Formel lautet:

$$f_{20}(L_i) := \frac{\sum_{L_j \in T(L_i)} \Delta(L_i, L_j)}{f_{19}(L_i)}.$$

Aus den bisherigen Überlegungen lässt sich überdies folgern, dass die meisten temporalen Nachbarn eines Text-Linienzugs andere Text-Linienzüge sind. Zuvor wurde

bereits gezeigt, dass Text-Linienzüge in der Regel kürzer sind als Form-Linienzüge. Hieraus folgt unmittelbar die Annahme, dass die durchschnittliche Länge der temporalen Nachbarn von Text-Linienzügen geringer ist als die durchschnittliche Länge der temporalen Nachbarn von Form-Linienzügen.

Spalte 3 in Tabelle 3.11 zeigt, dass temporale Nachbarn von Text-Linienzügen im Durchschnitt eine Länge von 35,44 Pixeln haben. Demgegenüber weisen temporale Nachbarn von Form-Linienzügen eine wesentlich größere durchschnittliche Länge von 103,88 Pixeln auf.

Die **durchschnittliche Länge der temporalen Nachbarn** eines Linienzugs L_i bestimmt sich, indem die Summe der Längen aller temporalen Nachbarn von L_i durch deren Anzahl dividiert wird:

$$f_{21}(L_i) := \frac{\sum_{L_j \in T(L_i)} f_1(L_j)}{f_{19}(L_i)}.$$

Räumliche Nachbarn

Linienzüge, die zu einem betrachteten Linienzug eine bestimmte maximale euklidische Distanz nicht überschreiten, werden im Folgenden als *räumliche Nachbarn* dieses Linienzugs bezeichnet. Verschiedene Forschergruppen schlagen vor, für die Betrachtung der räumlichen Nähe einen Grenzwert zwischen 4 und 10 Pixeln zu verwenden [59, 238]. Im Folgenden werden 10 Pixel als Grenzwert gewählt, weil hiermit bei der Analyse der entsprechenden Features die besten Ergebnisse erzielt werden konnten. Dementsprechend sei für einen Linienzug $L_i \in L(S)$ die Menge seiner räumlichen Nachbarn gegeben durch

$$R(L_i) := \{L_j \in L(S) \mid L_j \neq L_i \wedge \Delta(L_i, L_j) \leq 10\}.$$

Die Menge $R(L_i)$ enthält somit alle Linienzüge einer betrachteten Skizze, deren euklidische Distanz zu L_i 10 Pixel oder weniger beträgt.

Man überlegt sich, dass die meisten Zeichen eines Textes jeweils zwischen zwei anderen Zeichen stehen, zu denen sie eine verhältnismäßig geringe Distanz aufweisen.

Tabelle 3.12.: Kennzahlen der deskriptiven Statistik für räumliche Nachbarn von Linienzügen.

		Anzahl räumlicher Nachbarn	Durchschnittliche euklidische Distanz zu räumlichen Nachbarn in px	Durchschnittliche Länge der räumlichen Nachbarn in px
Texte	Min	0	0	0
	Max	11	10	1 295,99
	$\overline{x}$	2,77	4,89	46,55
	s	1,17	1,67	62,35
Formen	Min	0	0	0
	Max	68	10	2 477,51
	$\overline{x}$	3,31	4,33	185,67
	s	3,18	2,29	237,95

Darüber hinaus bestehen einige Zeichen aus mehr als einem Linienzug. Somit ist für Text-Linienzüge zu erwarten, dass diese jeweils zwei bis drei räumliche Nachbarn haben.

Für Formen kann ein etwas höherer Wert angenommen werden, weil diese häufig mit anderen Formen verbunden sind und deshalb zu diesen einen entsprechend niedrigen Abstand aufweisen. Zudem haben Formen oft eine verhältnismäßig geringe Distanz zu den Text-Linienzügen, die von ihnen umrandet werden.

Spalte 1 in Tabelle 3.12 zeigt, dass Text-Linienzüge durchschnittlich 2,77 räumliche Nachbarn mit einem Abstand $\leq$ 10 Pixel haben. Demgegenüber besitzen Form-Linienzügen im Durchschnitt 3,31 räumliche Nachbarn. Für Text-Linienzüge konnte ein Maximalwert von 11 beobachtet werden, während ein Form-Linienzug existiert, der sogar 68 räumliche Nachbarn besitzt. Hierbei handelt es sich um die Umrandung eines Textes, die einen sehr geringen Abstand zu den umfassten Text-Linienzügen aufweist.

Die **Anzahl räumlicher Nachbarn** eines Linienzugs L_i ist gegeben durch

$$f_{22}(L_i) := |R(L_i)|.$$

Weiterhin lässt sich beobachten, dass die durchschnittliche Distanz zu räumlichen Nachbarn für Text-Linienzüge höher ist als für Form-Linienzüge. Dies mag zunächst überraschen, wird aber anhand einiger einfacher Beispiele schnell nachvollziehbar: Viele Formen sind aus mehreren Form-Linienzügen zusammengesetzt. Diese schneiden sich entweder oder haben zueinander einen sehr geringen Abstand. Eine Pfeilspitze hängt beispielsweise in der Regel ohne oder mit nur einem sehr geringen Zwischenraum an ihrer zugehörigen Linie. Ebenso werden Rechtecke und Rauten, sofern sie aus mehreren Linienzügen bestehen, in der Regel so gezeichnet, dass möglichst keine oder nur geringe Lücken in den Außenlinien entstehen. Demgegenüber existiert zwischen den aufeinanderfolgenden Zeichen eines Wortes häufig eine größere räumliche Distanz – aufeinanderfolgende Zeichen schneiden oder berühren sich in der Regel nicht.

Spalte 2 in Tabelle 3.12 zeigt, dass Text-Linienzüge durchschnittlich eine euklidische Distanz von 4,89 Pixeln zu ihren räumlichen Nachbarn aufweisen. Demgegenüber haben Form-Linienzüge einen geringeren durchschnittlichen Abstand von 4,33 Pixeln zu ihren räumlich benachbarten Linienzügen.

Der **durchschnittliche euklidische Abstand zu räumlichen Nachbarn** berechnet sich für einen Linienzug L_i als Quotient aus der Summe der Abstände zu allen seinen Nachbarn und der Anzahl dieser:

$$f_{23}(L_i) := \frac{\sum_{L_j \in R(L_i)} \Delta(L_i, L_j)}{f_{22}(L_i)}.$$

Aus den bisherigen Überlegungen lässt sich folgern, dass die räumlichen Nachbarn von Text-Linienzügen – bei denen es sich oftmals selbst um Text-Linienzüge handelt – eine geringere durchschnittliche Länge aufweisen als die räumlichen Nachbarn von Form-Linienzügen.

Tabelle 3.12 bestätigt dies: Aus Spalte 3 ist ersichtlich, dass die durchschnittliche Länge räumlicher Nachbarn von Text-Linienzügen 46,55 Pixel beträgt. Demgegenüber sind räumliche Nachbarn von Form-Linienzügen im Durchschnitt 185,67 Pixel lang.

Die **durchschnittliche Länge der räumlichen Nachbarn** eines Linienzugs L_i bestimmt sich als Quotient der Summe der Längen aller räumlichen Nachbarn und der Anzahl dieser. Es ist

$$f_{24}(L_i) := \frac{\sum_{L_j \in R(L_i)} f_1(L_j)}{f_{22}(L_i)}.$$

3.4.6. Empirische Bewertung der Features

In den vorherigen Abschnitten wurden zwölf lokale und zwölf globale Features zur Klassifizierung von Linienzügen identifiziert, die für jeden Linienzug $L_i \in L(S)$ einen numerischen Vektor $f(L_i) := (f_1(L_i), \ldots, f_{24}(L_i)) \in \mathbb{R}^{24}$ ergeben.

Nachfolgend werden die Ergebnisse eines Experiments präsentiert, das die Qualität der identifizierten Features untersucht. Hierzu wurden wie eingangs beschrieben die in Abschnitt 3.3 vorgestellten digital rekonstruierten Interaction-Room-Projekte analysiert, welche insgesamt 48 617 Linienzüge enthalten.

Das Experiment wurde als k-fache Kreuzvalidierung durchgeführt. Die Grundidee dieses Verfahrens ist Folgende [251]: Eine betrachtete Datenmenge D, die aus n Elementen besteht, wird per Zufallsverfahren in $1 < k \leq n$ möglichst gleichgroße Teilmengen $D_1, \ldots, D_k$ zerlegt. Basierend auf diesen Teilmengen werden k Versuchsdurchläufe ausgeführt, wobei in jedem Durchlauf eine Teilmenge D_i mit $1 \leq i \leq k$ als Testmenge und die verbleibenden $k-1$ Teilmengen $\{D_1, \ldots, D_k\} \setminus D_i$ als Trainingsmengen verwendet werden. Das Gesamtergebnis des Experiments bestimmt sich am Ende als Durchschnittswert über alle k Einzeldurchläufe.

Der Vorteil einer Kreuzvalidierung liegt darin, dass jeder Datensatz sowohl zum Training als auch zum Test des Systems verwendet werden kann. Um die Varianz der erzielten Ergebnisse dabei so weit wie möglich zu minimieren, wird eine sogenannte Stratifikation der Daten durchgeführt [251]. Dies bedeutet, dass die zufällig bestimmten Teilmengen $D_1, \ldots, D_k$ so gewählt werden, dass sie eine möglichst gleichmäßige Verteilung der Daten aufweisen.

Tabelle 3.13.: Gleichmäßige Verteilung der Datenmenge auf zehn Teilmengen zur Durchführung einer 10-fachen stratifizierten Kreuzvalidierung.

	Trainingsdaten			Testdaten		
Teilmenge(n)	Text	Form	Gesamt	Text	Form	Gesamt
$D_1, \ldots, D_9$	39 258	4 497	43 755	4 362	500	4 862
D_{10}	39 258	4 500	43 758	4 362	497	4 859

Für das nachfolgende Experiment wird $k := 10$ gesetzt und somit eine 10-fache stratifizierte Kreuzvalidierung durchgeführt, weil Studien gezeigt haben, dass sich mit diesem Wert oft die besten Ergebnisse erzielen lassen [251].

Die Gesamtdatenmenge von 48 617 Linienzügen, bestehend aus 43 620 Text- und 4 997 Form-Linienzügen, wird zur Durchführung der zehn Durchläufe möglichst gleichmäßig auf zehn Teilmengen verteilt. Die resultierende Verteilung ist in Tabelle 3.13 dargestellt: Es ergeben sich neun Teilmengen $D_1, \ldots, D_9$ mit jeweils 43 755 Trainings- und 4 862 Testdatensätzen. Die Trainingsdaten unterteilen sich dabei jeweils in 39 258 Text- und 4 497 Form-Linienzüge. Die Testdaten unterteilen sich jeweils in 4 362 Text- und 500 Form-Linienzüge. Weil die Gesamtzahl der Form-Linienzüge nicht glatt durch 10 teilbar ist, weist die letzte Teilmenge D_{10} eine geringfügig andere Verteilung auf. Sie enthält 43 758 Trainings- und 4 859 Testdatensätze. Die Trainingsdaten unterteilen sich dabei in 39 258 Text- und 4 500 Form-Linienzüge, während die Testdaten 4 362 Text- und 497 Form-Linienzüge umfassen.

Bevor das Experiment durchgeführt werden kann, müssen die Zahlenwerte aller Features zunächst standardisiert werden, weil diese in stark unterschiedlichen Wertebereichen vorliegen. Hierdurch könnten einzelne Features aufgrund großer nomineller Werte das Endergebnis zu stark beeinflussen und somit verzerren. Dies kann durch eine Standardisierung der Werte verhindert werden, indem alle Wertebereiche gemäß der Standardnormalverteilung skaliert werden. Zur Standardisierung der Werte für einen Linienzug L_i wird die folgende gängige Formel verwendet [181]:

$$f_k(L_i)' := \frac{f_k(L_i) - \overline{x}_k}{s_k}.$$

Tabelle 3.14.: Übersicht über alle korrekt und falsch klassifizierten Linienzüge bei Durchführung einer 10-fachen Kreuzvalidierung.

	Text				Form			
	Korrekt		Falsch		Korrekt		Falsch	
Durchlauf	#	%	#	%	#	%	#	%
1	4 343	99,56	19	0,44	459	91,8	41	8,2
2	4 345	99,61	17	0,39	472	94,4	28	5,6
3	4 352	99,77	10	0,23	474	94,8	26	5,2
4	4 347	99,66	15	0,34	465	93	35	7
5	4 349	99,7	13	0,3	463	92,6	37	7,4
6	4 353	99,79	9	0,21	468	93,6	32	6,4
7	4 346	99,63	16	0,37	464	92,8	36	7,2
8	4 343	99,56	19	0,44	463	92,6	37	7,4
9	4 344	99,59	18	0,41	473	94,6	27	5,4
10	4 345	99,61	17	0,39	458	92,15	39	7,85
Ø	4 346,7	99,65	15,3	0,35	465,9	93,24	33,8	6,76

Gemäß Abschnitt 3.2.9 bezeichne $\overline{x}_k$ hierbei den Mittelwert aller Werte für das Feature f_k und s_k die entsprechende Standardabweichung. Die vorgenannte Formel zentriert demnach zunächst jedes Merkmal durch Subtraktion des Mittelwerts und skaliert es anschließend mit der empirischen Standardabweichung. Danach hat jedes Feature einen einheitlichen Mittelwert von 0 und eine Standardabweichung von 1. Auf diese Weise können die Eingabedaten vollständig unabhängig von Wertebereichen und Maßeinheiten analysiert werden.

Tabelle 3.14 zeigt die Ergebnisse der zehn Versuchsdurchläufe und listet Anzahl und Prozentanteil der jeweils korrekt und falsch klassifizierten Text- und Form-Linienzüge auf. Erwartungsgemäß unterscheiden sich die einzelnen Durchläufe dabei nur geringfügig. Insgesamt können im Durchschnitt 99,65 % aller Text-Linienzüge korrekt klassifiziert werden. Lediglich 0,35 % aller Text-Linienzüge werden durchschnittlich fälschlicherweise als Form-Linienzüge erkannt. Die Erkennungsrate für Form-Linienzüge fällt etwas geringer aus: Im Durchschnitt können 93,24 % aller Form-Linienzüge korrekt klassifiziert werden. 6,76 % werden vom Klassifizierer durchschnittlich fälschlicherweise für Text-Linienzüge gehalten.

Es fällt auf, dass die Erkennungsrate für Text-Linienzüge deutlich über der Erkennungsrate von Form-Linienzügen liegt. Es kann angenommen werden, dass dieser Umstand daher rührt, dass die zugrundeliegende Datenbasis wesentlich mehr Text-Linienzüge als Form-Linienzüge enthält. Das Machine-Learning-System ist deshalb darauf trainiert, dass Linienzüge in Freihandskizzen mit höherer Wahrscheinlichkeit eher Texte als Formen repräsentieren. Dies entspricht jedoch in der Regel auch tatsächlich der Realität, weil Texte wie eingangs diskutiert häufig aus vielen kurzen Linienzügen und Formen dagegen eher aus wenigen langen Linienzügen bestehen. Insbesondere Interaction-Room-Skizzen enthalten deshalb für gewöhnlich stets mehr Text- als Form-Linienzüge.

Insgesamt können die erzielten Ergebnisse damit als zufriedenstellend angesehen werden. Die 24 identifizierten Features eignen sich somit sehr gut zur Unterscheidung von Text- und Form-Linienzügen und können deshalb zur Klassifizierung im AugIR verwendet werden.

3.4.7. Zusammenfassung und Diskussion

In diesem Kapitel wurden zwölf lokale und zwölf globale Features zur Klassifizierung von Linienzügen vorgestellt. Für jeden Linienzug $L_i \in L(S)$ wird hieraus ein numerischer Vektor $f(L_i) := (f_1(L_i), \ldots, f_{24}(L_i)) \in \mathbb{R}^{24}$ berechnet. Dieser Vektor kann dann in einen zuvor trainierten Machine-Learning-Algorithmus eingegeben werden, der den zugehörigen Linienzug als Text- oder Form-Linienzug klassifiziert. Tabelle 3.15 fasst noch einmal alle vorgestellten lokalen und globalen Features zusammen.

Hierbei ist jedoch zu beachten, dass es sich bei den vorgestellten Features um Heuristiken handelt, die anhand einer gründlichen Analyse realer Interaction-Room-Skizzen ermittelt wurden. Aus diesem Grund treffen nicht immer alle Merkmale in gleicher Weise auf einen betrachteten Linienzug zu und es ist überdies stets mit einer gewissen Anzahl falsch klassifizierter Linienzüge zu rechnen. Die erreichte Qualität der Klassifizierung hängt unter anderem stark von der individuellen Handschrift eines Stakeholders ab. Je ordentlicher ein Stakeholder schreibt und je näher seine

Tabelle 3.15.: Übersicht über alle lokalen und globalen Features zur Klassifizierung von Linienzügen.

#	Typ	Beschreibung
f_1	lokal	Länge des Linienzugs
f_2	lokal	Dauer der Zeichenoperation
f_3	lokal	Anzahl der Kontrollpunkte
f_4	lokal	Anzahl der Selbstüberschneidungen
f_5	lokal	Euklidischer Abstand zwischen erstem und letztem Kontrollpunkt
f_6	lokal	Breite der Bounding Box
f_7	lokal	Höhe der Bounding Box
f_8	lokal	Flächeninhalt der Bounding Box
f_9	lokal	Diagonaler Durchmesser der Bounding Box
f_{10}	lokal	Winkel der Bounding-Box-Diagonale
f_{11}	lokal	Kosinus des initialen Winkels
f_{12}	lokal	Kosinus des Winkels zwischen erstem und letztem Kontrollpunkt
f_{13}	global	Anzahl umfassender Bounding Boxen
f_{14}	global	Anzahl vollständig enthaltener Linienzüge
f_{15}	global	Zeitlicher Abstand zum vorherigen Linienzug
f_{16}	global	Zeitlicher Abstand zum nachfolgenden Linienzug
f_{17}	global	Euklidischer Abstand zum vorherigen Linienzug
f_{18}	global	Euklidischer Abstand zum nachfolgenden Linienzug
f_{19}	global	Anzahl temporaler Nachbarn
f_{20}	global	Durchschnittlicher euklidischer Abstand zu temporalen Nachbarn
f_{21}	global	Durchschnittliche Länge der temporalen Nachbarn
f_{22}	global	Anzahl räumlicher Nachbarn
f_{23}	global	Durchschnittlicher euklidischer Abstand zu räumlichen Nachbarn
f_{24}	global	Durchschnittliche Länge der räumlichen Nachbarn

Schrift am definierten idealtypischen Schriftbild liegt, desto höher ist die Erkennungsrate für Text-Linienzüge.

Weiterhin spielt der Kontext eine große Rolle, in dem die Linienzüge gezeichnet werden. Obwohl Text-Linienzüge in der Regel kürzer sind als Form-Linienzüge, können beispielsweise Zeichen in der Überschrift einer Skizze wesentlich größer sein als darunter stehende Formen.

Überdies existieren bei bestimmten Merkmalen Ausnahmen für gewisse Zeichen. Beispielsweise wurde festgestellt, dass die meisten Zeichen von links nach rechts geschrieben werden. Für den Buchstaben „s“ gilt dies aber zum Beispiel nicht – er wird stattdessen in der Regel von rechts oben nach links unten gezeichnet.

Dennoch wurde im Rahmen eines abschließenden Experiments deutlich, dass sich die ausgewählten Features sehr gut zur Unterscheidung von Text- und Form-Linienzügen in den digital rekonstruierten Skizzen eignen. Durchschnittlich können mit ihnen 99,65 % aller Text-Linienzüge und 93,24 % aller Form-Linienzüge in einer Skizze korrekt klassifiziert werden.

3.5. Automatische Gruppierung von Linienzügen

Im vorherigen Abschnitt wurde erläutert, wie durch Betrachtung charakteristischer Eigenschaften Text- und Form-Linienzüge unterschieden werden können. Um auf den in dieser Weise identifizierten potentiellen Text-Linienzügen eine Handschrifterkennung durchführen zu können, müssen diese jedoch zunächst zu Zeichen, Wörtern und gegebenenfalls vollständigen Sätzen zusammengefasst werden. Dieser Prozess wird im Folgenden beschrieben.

3.5.1. Vorüberlegungen und Definitionen

Zur Gruppierung zusammengehöriger Linienzüge können ihr Erstellungszeitpunkt sowie ihre Position auf dem Whiteboard betrachtet werden. Dies ist eine gängige

Strategie, die von vielen verwandten Forschungsarbeiten verfolgt wird [59, 116, 238]. Linienzüge werden somit einer gemeinsamen logischen Gruppe zugeordnet, wenn sie etwa zur gleichen Zeit (zeitliche Nähe) und mit ausreichend geringem Abstand zueinander (räumliche Nähe) gezeichnet werden.

Verschiedene Forschergruppen schlagen übereinstimmend vor, für die Betrachtung der zeitlichen Differenz einen maximalen Abstand von 3,5 Sekunden zu wählen [59, 238].

Hinsichtlich der räumlichen Distanz schlagen unterschiedliche Ansätze einen maximalen Abstand zwischen 4 Pixeln [238] und 10 Pixeln [59] vor. Experimente mit einer prototypischen Implementierung des AugIRs haben gezeigt, dass eine Distanz von 10 Pixeln bei der Gruppierung zusammengehöriger Linienzüge in AugIR-Canvases die besten Ergebnisse erzielt, weshalb diese Grenze im Folgenden als Schwellwert gewählt wird.

Bevor ein entsprechender Algorithmus definiert werden kann, überlegt man sich zunächst, dass die Betrachtung des direkten Vorgängers eines Linienzugs allein nicht ausreichend ist:

Zum einen können im AugIR mehrere Stakeholder gleichzeitig in derselben Skizze arbeiten. Deshalb kann es vorkommen, dass parallel unterschiedliche Linienzüge zum gleichen Zeitpunkt von verschiedenen Stakeholdern gezeichnet werden. Hierdurch können mehrere Gruppen von Linienzügen entstehen, denen abwechselnd neue Linienzüge hinzugefügt werden müssen. Obwohl zwischen diesen Linienzügen geringe zeitliche Distanzen bestehen und sie in der geordneten Liste $L(S)$ aller Linienzüge direkt aufeinanderfolgen können, gehören sie dennoch häufig nicht zu der gleichen Gruppe.

Zum anderen gilt insbesondere für Texte, die sich über mehrere Zeilen erstrecken, dass der Abstand des letzten Zeichens einer Zeile i häufig einen großen euklidischen Abstand zum ersten Zeichen der darauffolgenden Zeile $i+1$ aufweist, weil Stakeholder für gewöhnlich jede Textzeile von links nach rechts schreiben. Demgegenüber ist jedoch der euklidische Abstand vom ersten Zeichen in Zeile i zum ersten Zeichen

in Zeile $i + 1$ in der Regel deutlich geringer, weil diese direkt untereinander oder zumindest nahe beisammen stehen.

Der zeitlich nächste Nachbar eines Linienzugs muss somit nicht gleichzeitig auch immer der Linienzug mit dem geringsten euklidischen Abstand sein. Aus diesem Grund reicht es nicht aus, lediglich den Abstand eines Linienzugs zu seinem direkten zeitlichen Vorgänger zu betrachten. Vielmehr müssen die Abstände zu allen Linienzügen aller zuvor erstellten Gruppen betrachtet werden, denen vor höchstens 3,5 Sekunden zuletzt ein Linienzug hinzugefügt wurde.

Um diesen Sachverhalt im nächsten Abschnitt mathematisch präzise formulieren zu können, werden im Folgenden einige Begriffe definiert:

Für eine betrachtete Skizze S bezeichne $G(S)$ die Menge aller Gruppen von Linienzügen, die in S vorkommen. Es ist somit $G(S) := (G_1, \ldots, G_m)$ mit Linienzuggruppen $G_1, \ldots, G_m$.

Jede Gruppe $G_i \in G(S)$ ist eine Menge von Linienzügen aus S, also $G_i := \{L_1, \ldots, L_n\}$ mit $L_j \in L(S)$ für $1 \leq j \leq n$.

Dabei ist jeder Linienzug $L_j \in L(S)$ stets immer in genau einer Gruppe enthalten. Für zwei Gruppen $G_i, G_k \in G(S)$ mit $i \neq k$ gilt demnach $G_i \cap G_k = \emptyset$.

Für einen Linienzug L bezeichne $NG(L)$ die Menge aller zeitlich und räumlich benachbarten Gruppen von L. Diese Menge ist formal gegeben durch

$$NG(L) := \{G_i \in G(S) \,|\, \exists L_j \in G_i : \Delta(L, L_j) \leq 10 \,\wedge\, \exists L_k \in G_i : \tau(L, L_k) \leq 3\,500\}.$$

Die Menge $NG(L)$ enthält also alle Gruppen aus $G(S)$, die einen Linienzug L_j enthalten, der zu L eine maximale euklidische Distanz von 10 Pixeln aufweist, sowie einen Linienzug L_k, der zu L eine maximale zeitliche Distanz von 3,5 Sekunden hat. Hierbei kann es sich selbstverständlich auch um den gleichen Linienzug handeln, es kann also $j = k$ sein.

Abschließend wird der euklidischen Abstand eines Linienzugs L zu einer Gruppe G definiert durch

$$\Delta(L, G) := \min\{\Delta(L, L_i) \,|\, L_i \in G\}.$$

$\Delta(L, G)$ beschreibt somit den minimalen Abstand des Linienzugs L zu allen Linienzügen der Gruppe G.

3.5.2. Algorithmus zur automatischen Gruppierung

Abbildung 3.8 illustriert den Algorithmus zur automatischen Gruppierung von Linienzügen in AugIR-Canvases.

Ein Linienzug gilt als fertig gezeichnet, sobald der zeichnende Stakeholder den Stift vom Board abhebt. Der entsprechende Linienzug wird daraufhin zunächst wie in Abschnitt 3.4 erläutert als Text- oder Form-Linienzug klassifiziert.

Als nächstes wird die zuvor eingeführte Menge $NG(L)$ bestimmt. Diese enthält alle zeitlich und räumlich benachbarten Gruppen von L, zu denen L potentiell als weiterer Linienzug hinzugefügt werden kann.

Falls die Menge $NG(L)$ leer ist, existiert noch keine geeignete Gruppe auf dem Canvas. Dies kommt beispielsweise dann vor, wenn die Erstellung des letzten Linienzugs vor L mehr als 3,5 Sekunden zurückliegt oder L an einer Stelle auf dem Whiteboard gezeichnet wurde, in deren näherer Umgebung keine passende Gruppe vorhanden ist. In diesem Fall wird eine neue Gruppe G_L erzeugt und L wird dieser Gruppe als einziges Element hinzugefügt. Anschließend wird die neu erstellte Gruppe G_L der Menge $G(S)$ hinzugefügt und die Gruppierung ist abgeschlossen.

Falls $NG(L)$ nicht leer ist, wird die Anzahl der Elemente in dieser Menge betrachtet:

Enthält $NG(L)$ lediglich eine Gruppe, so ist diese eindeutig bestimmt und wird als G_L ausgewählt.

Enthält $NG(L)$ mehr als eine Gruppe, so wird diejenige Gruppe als G_L ausgewählt, die den kleinsten euklidischen Abstand zu L aufweist. Dies ist nur in seltenen Ausnahmefällen notwendig, wenn mehrere Stakeholder zur gleichen Zeit dicht nebeneinander auf das Whiteboard schreiben.

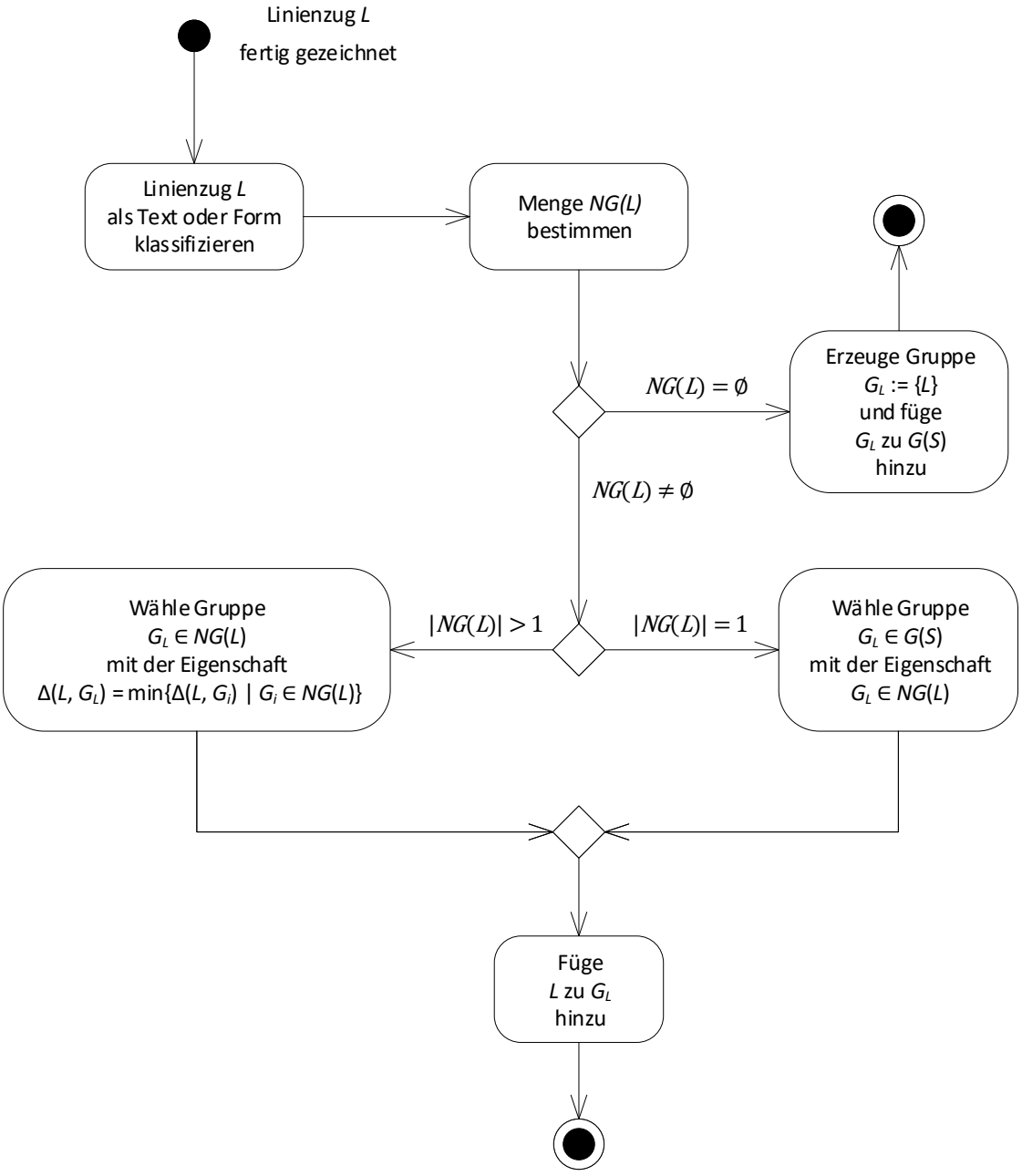

Abbildung 3.8.: Algorithmus zur automatischen Gruppierung von Linienzügen.

Im letzten Schritt wird der Linienzug L der zuvor ausgewählten Gruppe G_L hinzugefügt. Damit ist die Gruppierung von L abgeschlossen.

Es sei angemerkt, dass der zuvor beschriebene Algorithmus auch Text- und Form-Linienzüge einer gemeinsamen Gruppe zuordnet, sofern diese einen ausreichend geringen zeitlichen und räumlichen Abstand zueinander aufweisen.

Man überlegt sich jedoch, dass diese Gruppierung nicht problematisch ist, weil anhand der zuvor durchgeführten Klassifizierung zwischen Text- und Form-Linienzügen unterschieden werden kann. Anstatt alle Linienzüge einer Gruppe in die Handschrifterkennung einzugeben, kann eine Vorauswahl erfolgen, so dass nur potentielle Text-Linienzüge weiter analysiert werden. Form-Linienzüge werden im Folgenden zunächst nicht weiter betrachtet, könnten aber beispielsweise im Rahmen zukünftiger Erweiterungen durch eine Mustererkennung analysiert werden, um die Semantik von Interaction-Room-Canvases noch genauer zu erfassen.

Ganz im Gegenteil ist das zuvor beschriebene Verhalten sogar gewünscht, weil hierdurch logisch zusammengehörige Linienzüge unabhängig von ihrer Klassifizierung zu einer Gruppe zusammengefasst werden. Ein Beispiel hierfür sind Aktionen in Prozessskizzen, die sowohl aus einer Beschriftung (Text-Linienzüge) als auch aus einem umgebenden Rahmen (Form-Linienzüge) bestehen. Diese Elemente können somit auf dem elektronischen Whiteboard als zusammengehörige Einheiten verschoben, bearbeitet, annotiert und über Trace Links miteinander verknüpft werden.

3.6. Tokenisierung von Texten

Wie in den vorangegangenen Abschnitten beschrieben, werden alle gezeichneten Linienzüge als Text- oder Form-Linienzüge klassifiziert und anschließend automatisch nach potentieller Zusammengehörigkeit gruppiert, um auf ihnen in geeigneter Weise eine Handschrifterkennung durchführen zu können.

Nachdem die Handschrifterkennung für alle Text-Linienzüge erfolgt ist, müssen die erkannten Texte im nächsten Schritt in Tupel aufgetrennt werden, die jeweils ein

oder mehrere Elemente umfassen. In der digitalen Sprachverarbeitung nennt man diesen Prozess *Tokenisierung* [233].

Tokenisierung ist ein wichtiger Schritt bei der Analyse von handschriftlichen Texten, um diese für eine anschließende Weiterverarbeitung vorzubereiten. Es entstehen hierbei Elemente aus $\mathcal{A}^{*^m}$, die beispielsweise als Eingaben für die Berechnung der in Abschnitt 3.2.8 definierten Kosinus-Ähnlichkeit dienen können. Das Ergebnis der Tokenisierung wird dabei maßgeblich von den Regeln bestimmt, die festlegen, wie genau ein Text aufzutrennen ist. Eine sehr naheliegende und häufig verwendete Regel ist beispielsweise eine Trennung des Textes an Leerzeichen. Jedes Element des durch die Tokenisierung erzeugten Tupels entspricht dann genau einem Wort des erkannten Textes.

Die vorgenannte Regel ließe sich beispielsweise für eine Zeichenkette $s \in \mathcal{A}^*$ als Abbildung $t : \mathcal{A}^* \to \mathcal{A}^{*^m}$ definieren durch

$$t(s) := \text{Split}(s, \text{„ “}).$$

Der Text

$$s := \text{„Ein beispielhafter Text“}$$

würde dann durch Anwendung der Regel t beispielsweise zur Tokenisierung

$$t(s) = (\text{„Ein“}, \text{„beispielhafter“}, \text{„Text“})$$

führen. Darüber hinaus sind jedoch auch beliebige andere Regeln denkbar, die jeweils in völlig unterschiedlichen Tokenisierungen resultieren können.

Auch im AugIR ist eine Tokenisierung von textuellen Elementen notwendig, damit diese später sinnvoll weiterverarbeitet und zum Beispiel auf ihre Ähnlichkeit hin untersucht werden können. Es stellt sich somit unweigerlich die Frage nach passenden Tokenisierungsregeln.

Eine sinnvolle und naheliegende Regel ist wie zuvor beschrieben die Trennung der Texte an Leerzeichen. Der folgende Abschnitt wird jedoch illustrieren, dass dies alleine zu einem unbefriedigenden Ergebnis führen kann. Es soll daher zunächst

diskutiert werden, welche grundsätzlichen Probleme bei der Erkennung von handschriftlichen Texten bestehen, damit aus diesen Erkenntnissen geeignete Tokenisierungsregeln abgeleitet werden können, die diesen Problemen entgegenwirken.

3.6.1. Analyse von Problemen bei der Handschrifterkennung

Zur Identifikation von potentiellen Problemen, die bei der Erkennung von handschriftlich geschriebenen Texten auftreten können, wurden die in Abschnitt 3.3 beschriebenen digital rekonstruierten Interaction-Room-Projekte analysiert. Die erkannten digitalen Texte wurden hierzu manuell mit den Inhalten der originalen Freihandzeichnungen abgeglichen. Folgende Probleme konnten dabei identifiziert werden:

Problem 1: Erkennung von Leerzeichen

Wenn ein Text aus mehreren Wörtern besteht und somit Leerzeichen enthält, hat die Handschrifterkennung gelegentlich Schwierigkeiten bei der akkuraten Erkennung aller Leerzeichen. Hierbei kann es vorkommen, dass fälschlicherweise zusätzliche Leerzeichen eingefügt werden, beispielsweise wenn beim Schreiben des Textes auf dem digitalen Whiteboard zwischen zwei Buchstaben eines Wortes ein zu großer Freiraum gelassen wurde. Umgekehrt kann es ebenfalls passieren, dass vorhandene Leerzeichen fälschlicherweise nicht erkannt und deshalb entfernt werden, weil zwei Wörter zu eng aneinander geschrieben wurden.

Dieses Problem kann zusätzlich dadurch verstärkt werden, dass der entsprechende Text Wörter enthält, die gegebenenfalls nicht in dem zugrundeliegenden Wörterbuch des Handschrifterkenners vorhanden sind. Für solche Wörter kann die Handschrifterkennung dann nämlich nicht entscheiden, ob sie gemäß der gültigen Rechtschreibregeln zusammen oder auseinander geschrieben werden müssen.

An einem einfachen Beispiel lässt sich leicht illustrieren, dass dieses Problem auch schon bei simplen Wörtern auftreten kann: Angenommen, zwei Stakeholder schreiben jeweils den Text „gegensteuern“. Im ersten Fall erkennt die Handschrifterken-

nung diesen Text fehlerfrei. Im zweiten Fall hat der Stakeholder jedoch zwischen den Buchstaben „n“ und „s“ einen zu großen Freiraum gelassen, weshalb die Handschrifterkennung den Text in die beiden separaten Wörter „gegen“ und „steuern“ zerlegt. Gemäß des zugrundeliegenden Wörterbuchs wären beide Varianten zunächst vollkommen korrekt, weil sowohl das Wort „gegensteuern“ als auch die Wörter „gegen“ und „steuern“ zulässige Wörter der deutschen Sprache sind.

Eine solche fehlerhafte Erkennung von Leerzeichen ist allerdings deshalb problematisch, weil sich hierdurch die Anzahl der Wörter im Text und somit häufig auch die Anzahl der Elemente in den durch die Tokenisierung erzeugten Tupeln ändern kann. Dies wirkt sich unmittelbar auf die Berechnung der Kosinus-Ähnlichkeit aus, wie man sich mit Hilfe der Definitionen aus Abschnitt 3.2.8 klar machen kann: Die zugrundeliegende Menge W aller Wörter bestünde in dem vorgenannten Beispiel aus den Elementen „gegensteuern“, „gegen“ und „steuern“. Es wäre also

$$W := \{\text{„gegensteuern“}, \text{„gegen“}, \text{„steuern“}\}.$$

Der Häufigkeitsvektor für das erste Dokument, das nur aus dem Wort „gegensteuern“ besteht, würde sich zu $q = (1, 0, 0)$ ergeben. Der Häufigkeitsvektor für das zweite Dokument, das aus den beiden Wörtern „gegen“ und „steuern“ besteht, würde sich jedoch zu $d = (0, 1, 1)$ ergeben.

Man rechnet nun leicht nach, dass

$$\text{CosSim}(q, d) = \frac{q \cdot d}{|q| \cdot |d|} = \frac{(1, 0, 0) \cdot (0, 1, 1)}{|(1, 0, 0)| \cdot |(0, 1, 1)|} = \frac{0}{\sqrt{1} \cdot \sqrt{2}} = 0$$

ist. Obwohl die Dokumente also eigentlich identisch sein sollten, ist ihre errechnete Kosinus-Ähnlichkeit 0. Um diesem Problem adäquat zu begegnen, sind folglich zusätzlich zur Trennung der Texte an Leerzeichen weitere Tokenisierungsregeln notwendig.

Problem 2: Erkennung von Sonderzeichen

Zusätzlich zur potentiell fehlerhaften Erkennung von Leerzeichen hat sich gezeigt, dass die Handschrifterkennung gelegentlich bei der zuverlässigen Erkennung von Sonderzeichen versagt, also bei der Erkennung von Elementen aus $\mathcal{S}$. Insbesondere sehr komplexe Zeichen, die schwierig zu malen sind, wie beispielsweise „§“ und „@“, werden gelegentlich falsch oder sogar gar nicht erkannt. In vielen Fällen wird dann ein Sonderzeichen vom Handschrifterkenner entweder durch ein ähnliches (aber falsches) Zeichen ersetzt oder es wird komplett verworfen. Dieses Problem trifft überdies auch auf Satztrennzeichen wie „.“ und „,“ zu.

Erschwerend kommt hinzu, dass die Handschrifterkennung insbesondere im Kontext von Sonderzeichen und Satztrennzeichen Probleme bei der eingangs bereits diskutierten Erkennung von Leerzeichen hat. Dies schließt beispielsweise häufig verwendete Sonderzeichen wie das Prozentzeichen, den Bindestrich und den Schrägstrich ein. Häufig ignoriert die Handschrifterkennung Leerzeichen, die vor oder nach einem solchen Zeichen stehen, oder fügt fälschlicherweise dort ein Leerzeichen hinzu.

In den analysierten Projekten hat die Handschrifterkennung zum Beispiel aus dem handschriftlichen Text „Contracts, customers“ fälschlich den Text „Contracts / Customers“ erkannt. Die Wortmenge W ergibt sich folglich zu

$$W := \{\text{„Contracts,“}, \text{„customers“}, \text{„Contracts“}, \text{„/“}, \text{„Customers“}\}.$$

Entsprechend sind die Häufigkeitsvektoren q und d gegeben durch $q = (1, 1, 0, 0, 0)$ und $d = (0, 0, 1, 1, 1)$. Dies führt zu einer Kosinus-Ähnlichkeit von $\text{CosSim}(q, d) = 0$, obwohl die Texte eigentlich identisch sein sollten und nur aufgrund eines geringfügigen Erkennungsfehlers voneinander abweichen.

Darüber hinaus wird an diesem Beispiel auch deutlich, dass es sinnvoll sein kann, alle Buchstaben der erkannten Texte einheitlich in Klein- oder Großbuchstaben umzuwandeln. Im obigen Beispiel werden die Wörter „customers“ und „Customers“ bei der Berechnung der Kosinus-Ähnlichkeit als zwei verschiedene Wörter betrachtet, obwohl sie sich lediglich in der Groß-/Kleinschreibung unterscheiden. Würde man

die Texte vorher vollständig in Kleinbuchstaben umwandeln, würde sich stattdessen folgende Wortmenge W mit vier statt fünf Elementen ergeben:

$$W := \{\text{„contracts,“}, \text{„customers“}, \text{„contracts“}, \text{„/“}\}.$$

Die resultierenden Häufigkeitsvektoren ergäben sich dann dementsprechend zu $q = (1, 1, 0, 0)$ und $d = (0, 1, 1, 1)$, was zu einer wesentlich höheren Kosinus-Ähnlichkeit von $\text{CosSim}(q, d) = 0{,}4082$ führen würde.

Später wird jedoch noch deutlich werden, dass 0,4082 immer noch ein sehr niedriger Ähnlichkeitswert ist. Eine Umwandlung aller Buchstaben in Kleinbuchstaben reicht also alleine nicht aus, um das vorgenannte Problem zu lösen.

Es kann somit insgesamt festgehalten werden, dass die zuverlässige Erkennung von Sonderzeichen – insbesondere wenn diese komplex zu zeichnen sind oder im Kontext von Leerzeichen stehen – eine besondere Schwierigkeit für den Handschrifterkenner darstellt. Auch dieser Aspekt muss bei der Wahl geeigneter Tokenisierungsregeln berücksichtigt werden.

Problem 3: Zeilenumbrüche und Erkennung von Worttrennungen

Manche Texte werden aufgrund ihrer Länge nicht in eine einzelne Zeile geschrieben sondern erstrecken sich gegebenenfalls über zwei oder mehr Zeilen. Dies trifft insbesondere auf Canvases zu, deren Elemente häufig längere Texte enthalten, wie beispielsweise Story Cards im Feature Canvas oder Aktionen im Process Canvas.

Gelegentlich kommt es vor, dass Stakeholder auf diese Weise auch längere Wörter über Zeilengrenzen hinweg auftrennen. Beispielsweise könnte der Text „Softwareentwicklung starten“ auf drei Zeilen aufgeteilt und geschrieben werden als

Zeile 1: „Software-“

Zeile 2: „entwicklung“

Zeile 3: „starten“

Das Wort „Softwareentwicklung“ erstreckt sich also über zwei Zeilen und enthält insbesondere einen Bindestrich, der beide Teile des Wortes grammatikalisch korrekt miteinander verbindet. Die Handschrifterkennung würde dieses Textelement in die drei Wörter „Software-“, „entwicklung“ und „starten“ übersetzen.

Wenn nun an anderer Stelle der gleiche Text in eine einzige Zeile geschrieben oder zumindest das Wort „Softwareentwicklung“ nicht getrennt würde, dann ergäbe sich für die Berechnung der Kosinus-Ähnlichkeit die Wortmenge W als

$$W := \{\text{„Software-“}, \text{„entwicklung“}, \text{„starten“}, \text{„Softwareentwicklung“}\}.$$

Die entsprechenden Häufigkeitsvektoren würden sich folglich zu $q = (1, 1, 1, 0)$ und $d = (0, 0, 1, 1)$ bestimmen, was zu einer Kosinus-Ähnlichkeit von $\text{CosSim}(q, d) = 0{,}4082$ führen würde.

Wie zuvor beschrieben ist dies ein sehr niedriger Ähnlichkeitswert, obwohl die ursprünglichen Texte eigentlich inhaltlich vollständig identisch sind. Es wird folglich auch eine passende Tokenisierungsregel benötigt, um mehrzeilige Texte und durch einen Bindestrich getrennte Wörter adäquat zu behandeln.

Darüber hinaus beherrschen nicht alle Stakeholder eine fehlerfreie Grammatik und Rechtschreibung. Hierdurch kann es vorkommen dass das Problem der Erkennung von Worttrennungen nicht nur bei mehrzeiligen Texten auftritt, sondern bereits bei Textelementen beobachtet werden kann, die nur aus einem einzigen Wort bestehen. Beispielsweise wird das Wort „Softwareentwicklung“ laut Duden zusammengeschrieben [65]. Häufig sieht man jedoch auch die (falsche) Schreibweise „Software-Entwicklung“. Weil in Interaction-Room-Workshops jedoch die Diskussionen und der kreative Austausch im Vordergrund stehen, werden korrekte Schreibweisen oft vernachlässigt und solche Fehler bleiben häufig unkorrigiert. Dies führt schließlich zu dem zuvor beschriebenen Problem, dass sich aus eigentlich identischen textuellen Inhalten aufgrund abweichender Schreibweisen unterschiedliche Häufigkeitsvektoren ergeben können, die fälschlicherweise in einer zu niedrigen Kosinus-Ähnlichkeit resultieren.

Problem 4: Erkennung von zusammengesetzten Wörtern

In vielen Sprachen werden zusammengesetzte Wörter gebildet, indem man einzelne Wörter mit einem Bindestrich miteinander verbindet. Anders als beim zuvor beschriebenen Problem stehen die Wörter hierbei in der Regel in der gleichen Zeile und sind auch grammatikalisch völlig korrekt. Jedoch können einzelne Bestandteile dieser zusammengesetzten Wörter auch als eigenständige Wörter in anderen Skizzen vorkommen und für sich alleine betrachtet Sinn ergeben.

In einer ersten Skizze könnte beispielsweise der Text „Prüfung ok“ stehen, während eine zweite Skizze den Text „Info-Prüfung“ enthält. Zwischen beiden Elementen besteht aufgrund des zentralen Wortes „Prüfung“ ein potentieller inhaltlicher Zusammenhang. Die Kosinus-Ähnlichkeit zwischen ihnen ist jedoch 0, was man leicht erkennt, wenn man die Wortmenge W betrachtet. Diese ist gegeben durch

$$W := \{\text{„Prüfung“}, \text{„ok“}, \text{„Info-Prüfung“}\}.$$

Die zugehörigen Häufigkeitsvektoren ergeben sich demnach zu $q = (1, 1, 0)$ und $d = (0, 0, 1)$. Dies führt zu einer Kosinus-Ähnlichkeit von $\text{CosSim}(q, d) = 0$, obwohl eine inhaltliche Verbindung aufgrund des gemeinsamen Wortes „Prüfung“ nicht unwahrscheinlich ist.

Damit auch solche Zusammenhänge erkannt werden können, muss eine Tokenisierungsregel existieren, die zusammengesetzte Wörter in geeigneter Weise in ihre Bestandteile auftrennt.

Problem 5: Länderspezifische Grammatikregeln und Schreibweisen

Zuletzt kommt noch das Problem hinzu, dass sich bestimmte Grammatikregeln und Schreibweisen in verschiedenen Ländern stark voneinander unterscheiden können.

So sieht beispielsweise das Deutsche Institut für Normung (DIN) vor, dass bei einer Prozentangabe im Deutschen zwischen der Zahl und dem Prozentzeichen ein Leerzeichen stehen muss [63]. In anderen Sprachen ist dies jedoch nicht immer so.

Im Englischen ist es beispielsweise üblich, das Prozentzeichen ohne Freiraum direkt hinter die zugehörige Zahl zu schreiben [167]. Während also in einem deutschen Text beispielsweise die Angabe „hundert Prozent“ als „100 %“ geschrieben werden würde, würde in einem englischen Text stattdessen „100%“ stehen.

Dies mag zunächst noch nicht problematisch erscheinen, jedoch steigt die geografisch verteilte Entwicklung von Softwaresystemen durch die zunehmende Vereinfachung von Kommunikationskanälen kontinuierlich an [67]. Es ist deshalb insbesondere in heterogenen Projektteams nicht unüblich, dass Stakeholder unterschiedlicher Herkunft und mit gegebenenfalls unterschiedlichen Muttersprachen zusammenarbeiten. In solchen Fällen können derart länderspezifische Schreibweisen zu Komplikationen führen, wenn die gleichen Sachverhalte an verschiedenen Stellen auf unterschiedliche Weisen notiert werden. Während die Handschrifterkennung im vorgenannten Beispiel aus dem Text „100 %“ zwei Wörter erkennen würde, wäre der Text „100%“ lediglich ein Wort. Es würden sich somit unterschiedliche Häufigkeitsvektoren für inhaltlich eigentlich identische Texte ergeben.

Ebenso wie die Existenz bestimmter länderspezifischer Regeln zu Problemen in der Kommunikation zwischen Stakeholdern verschiedener Herkunftsländer führen kann, so kann das Fehlen von Regeln ein Problem innerhalb einer Sprachgemeinschaft darstellen.

Beispielsweise herrscht in der deutschen Sprache keine einheitliche Auffassung darüber, ob vor oder nach einem Schrägstrich ein Leerzeichen zu stehen hat. Die Norm DIN 5008 [63], die Schreib- und Gestaltungsregeln für die Textverarbeitung festlegt, gibt zwar an, dass vor und nach einem Schrägstrich eigentlich kein Leerzeichen stehen darf, bezieht sich hierbei jedoch nur auf kurze und ungegliederte Ausdrücke. Es bleibt somit unklar, ob diese Regel auch auf längere und mehrteilige Ausdrücke anzuwenden ist. Insbesondere, wenn auf mindestens einer Seite eines Schrägstrichs kein einzelnes Wort sondern eine Wortgruppe steht, kann die Lesbarkeit erhöht werden, indem vor und hinter dem Schrägstrich ein Leerzeichen gesetzt wird. Aufgrund dieser Ungewissheit könnten manche Stakeholder beispielsweise „und/oder“ schreiben, während andere den gleichen Text an anderer Stelle innerhalb eines Projekts

als „und / oder“ notieren. Während ersteres von der Handschrifterkennung als ein einzelnes Wort erkannt wird, wird die zweite Variante in drei Wörter übersetzt.

Die vorangegangen Beispiele illustrieren, dass länderspezifische Schreibweisen und Grammatikregeln insbesondere im Zusammenhang mit Leer- und bestimmten Sonderzeichen zu Problemen führen können. Dieser Aspekt muss ebenfalls bei der Wahl geeigneter Tokenisierungsregeln berücksichtigt werden.

3.6.2. Definition von heuristischen Regeln zur Tokenisierung

Basierend auf den im vorherigen Abschnitt diskutierten Beobachtungen und Erkenntnissen wurden drei heuristische Regeln zur Tokenisierung entwickelt. Diese werden im Folgenden definiert und kommen später bei der Erfassung der Trace Links zur Anwendung.

Die Grundidee besteht darin, auf jedes Textelement alle drei Regeln anzuwenden und hierdurch mehrere – im Allgemeinen verschiedene – Tokenisierungen für jedes Dokument zu erzeugen. Hierzu werden im Folgenden drei Tokenisierungsregeln t_1, t_2 und t_3 definiert. Durch Anwendung dieser Regeln auf einen Text $s \in \mathcal{A}^*$ entstehen drei Tupel von Wörtern $t_1(s), t_2(s), t_3(s) \in \mathcal{A}^{*^m}$. Anstatt zur Prüfung der Ähnlichkeit zwischen s und einem anderen Textelement $u \in \mathcal{A}^*$ die Elemente s und u direkt miteinander zu vergleichen, werden ihre Tokenisierungen $t_i(s)$ und $t_i(u)$ für $1 \leq i \leq 3$ mit Hilfe der in Abschnitt 3.2.8 eingeführten Kosinus-Ähnlichkeit miteinander verglichen.

Dieses Vorgehen sorgt dafür, dass viele der vorgenannten Probleme behoben und Zusammenhänge erkannt werden können, die ansonsten übersehen worden wären. In naheliegender Weise entstehen hierdurch aber auch falsche Treffer, indem Dokumente als ähnlich bewertet werden, die eigentlich verschieden sind. Die Ähnlichkeit kommt in solchen Fällen ausschließlich aufgrund der durchgeführten Tokenisierungen zustande. Dieser Nachteil wird jedoch durch die zusätzlich gefundenen Zusammenhänge ausgeglichen, wie die Evaluation der Trace-Link-Erfassung in Kapitel 7 zeigen wird.

Als erste Tokenisierungsregel wird in naheliegender Weise die eingangs vorgestellte Regel verwendet, die einen Text an seinen Leerzeichen auftrennt und dadurch jedem Element des durch die Tokenisierung erzeugten Tupels genau ein Wort des erkannten Textes zuweist. Darüber hinaus wird für diese sowie für alle folgenden Regeln vereinbart, dass bei der Tokenisierung wie in Abschnitt 3.6.1 diskutiert stets alle Buchstaben in Kleinbuchstaben umgewandelt werden.

Die vorgenannte Regel lässt sich somit als Abbildung $t_1 : \mathcal{A}^* \rightarrow \mathcal{A}^{*^m}$ definieren durch

$$t_1(s) := \text{Split}^+(\text{ToLower}(s), \beta).$$

Aus dem Text

$$s := \text{„Ein beispielhafter Text“}$$

würde sich dann durch Anwendung der Regel t_1 beispielsweise das Tupel

$$t_1(s) = (\text{„ein“}, \text{„beispielhafter“}, \text{„text“})$$

ergeben.

Diese Tokenisierung führt immer dann zum Erfolg, wenn die handschriftlichen Texte korrekt erkannt wurden oder nur marginale Fehler enthalten. Insbesondere für die Verknüpfung von Texten, die nur aus wenigen Wörtern bestehen und keine komplexen Sonderzeichen enthalten, ist diese Regel gut geeignet und zumeist ausreichend.

Die zuvor beschriebene fehleranfällige Erkennung von Leerzeichen – insbesondere im Kontext von Sonderzeichen – lässt sich signifikant dadurch verbessern, dass alle Leer- und Sonderzeichen bei der Tokenisierung vollständig aus dem Text entfernt werden. Alle Wörter der betrachteten Zeichenkette werden somit zusammengefügt und es ergibt sich ein Tupel mit genau einem einzigen Element. Diese Tokenisierungsregel kann als Abbildung $t_2 : \mathcal{A}^* \rightarrow \mathcal{A}^{*^m}$ definiert werden durch

$$t_2(s) := (\text{RemoveAll}(\text{ToLower}(s), \mathcal{S})).$$

Sie behebt gleichzeitig mehrere Probleme:

Die fehleranfällige Erkennung von Leerzeichen wird durch diese Regel vollständig eliminiert, weil alle Leerzeichen aus der Zeichenkette entfernt werden. Wenn die Handschrifterkennung also fälschlicherweise zusätzliche Leerzeichen einfügt oder vorhandene Leerzeichen nicht erkennt, wirkt sich dies nicht auf die erzeugten Tupel aus. Beispielsweise ergibt sich für die beiden Dokumente „gegensteuern“ und „gegen steuern“ unter Anwendung dieser Tokenisierungsregel jeweils die gleiche Tokenisierung

$$t_2(\text{„gegensteuern“}) = (\text{„gegensteuern“}) = t_2(\text{„gegen steuern“}).$$

Die resultierenden Häufigkeitsvektoren sind folglich identisch und deren Kosinus-Ähnlichkeit berechnet sich richtigerweise zu 1.

Mit dieser Regel lässt sich darüber hinaus auch die problematische Erkennung von Sonderzeichen signifikant verbessern. Betrachtet man hierzu beispielsweise die beiden Dokumente „Contracts, customers“ und „Contracts / Customers“, so ergibt sich auch hierfür jeweils die gleiche Tokenisierung

$$\begin{aligned} t_2(\text{„Contracts, customers“}) &= (\text{„contractscustomers“}) \\ &= t_2(\text{„Contracts / Customers“}), \end{aligned}$$

weil alle Leerzeichen sowie die beiden Sonderzeichen „,“ und „/“ entfernt werden.

Zum Teil kann auch das Problem der Erkennung von Zeilenumbrüchen und Worttrennungen durch Anwendung der Regel t_2 deutlich gemindert werden. Beispielsweise ergibt sich auch für die beiden Dokumente „Software- entwicklung starten“ und „Softwareentwicklung starten“ eine identische Tokenisierung mit

$$\begin{aligned} t_2(\text{„Software- entwicklung starten“}) &= (\text{„softwareentwicklungstarten“}) \\ &= t_2(\text{„Softwareentwicklung starten“}). \end{aligned}$$

Diese Tokenisierungsregel ist auch für die Lösung bestimmter Probleme durch länderspezifische Grammatikregeln und Schreibweisen geeignet, wie man sich leicht durch Betrachtung der in Abschnitt 3.6.1 genannten Beispiele überlegen kann. Für die Angabe „100 Prozent“ ergäbe sich beispielsweise sowohl für „100 %“ als auch

für „100%“ jeweils die gleiche Tokenisierung

$$t_2(\text{„100 %“}) = (100) = t_2(\text{„100%“}).$$

An diesen Beispielen wird jedoch auch eine bereits zuvor erwähnte Problematik deutlich: Durch die Entfernung sämtlicher Leer- und Sonderzeichen geht ebenfalls ein Teil der Information des Textes verloren. Nach Durchführung der Tokenisierung kann beispielsweise nicht mehr zwischen der Angabe „100 %“ und der Zahl „100“ unterschieden werden. Dies kann mitunter zu falschen Verknüpfungen führen.

Darüber hinaus ist anzumerken, dass mit der vorgenannten Regel selbstverständlich nicht pauschal alle Probleme hinsichtlich länderspezifischer Grammatik- und Rechtschreibregeln gelöst werden können, weil deren Anzahl für existierende Sprachen unüberschaubar groß ist. Für die zuvor genannten Beispiele und ähnliche Szenarien löst sie die identifizierten Probleme jedoch zufriedenstellend.

Obwohl die Tokenisierungsregel t_2 bereits zahlreiche Schwierigkeiten beseitigt, können noch nicht alle in Abschnitt 3.6.1 genannten Probleme durch sie behoben werden. Für Dokumente, in denen zusammengesetzte Wörter vorkommen, die durch eine Tokenisierung getrennt werden müssen, wird eine zusätzliche Regel benötigt, was man sich wie folgt überlegt:

Für die Texte „Prüfung ok“ und „Info-Prüfung“ würden sich unter Anwendung der Regel t_2 die beiden Tokenisierungen

$$t_2(\text{„Prüfung ok“}) = (\text{„prüfungok“})$$

und

$$t_2(\text{„Info-Prüfung“}) = (\text{„infoprüfung“}).$$

ergeben. Offensichtlich sind diese Tokenisierungen komplett verschieden, was zu einer Kosinus-Ähnlichkeit von 0 führen würde.

Es erscheint deshalb sinnvoll, eine zusätzliche Tokenisierungsregel zu definieren, die alle Sonderzeichen eines Textes durch Leerzeichen ersetzt und anschließend den Text

an Leerzeichen trennt. Hierbei handelt es sich also um eine Kombination der Regeln t_1 und t_2, die sich als Abbildung $t_3 : \mathcal{A}^* \to \mathcal{A}^{*^m}$ mit

$$t_3(s) := \text{Split}^+(\text{ReplaceAll}(\text{ToLower}(s), \mathcal{S} \setminus \beta, \beta), \beta)$$

definieren lässt. Mit dieser Tokenisierungsregel lassen sich Probleme bei der Erkennung von zusammengesetzten Wörtern lösen, wie das folgende Beispiel illustriert:

Für die Texte „Prüfung ok“ und „Info-Prüfung“ ergeben sich unter Anwendung der Regel t_3 die beiden folgenden Tokenisierungen

$$t_3(\text{„Prüfung ok“}) = (\text{„prüfung ok“})$$

und

$$t_3(\text{„Info-Prüfung“}) = (\text{„info“}, \text{„prüfung“}).$$

Aus diesen Tokenisierungen kann nun die zugrundeliegende Menge W aller Wörter bestimmt werden als

$$W := \{\text{„prüfung“}, \text{„ok“}, \text{„info“}\}.$$

Der Häufigkeitsvektor q für das erste Dokument ergibt sich entsprechend zu $q = (1, 1, 0)$, während sich der Häufigkeitsvektor d für das zweite Dokument zu $d = (1, 0, 1)$ ergibt. Die Kosinus-Ähnlichkeit lässt sich somit berechnen als

$$\text{CosSim}(q, d) = \frac{(1, 1, 0) \cdot (1, 0, 1)}{|(1, 1, 0)| \cdot |(1, 0, 1)|} = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = 0{,}5.$$

Es liegt in diesem Fall also eine mittelgroße Ähnlichkeit aufgrund des gemeinsamen Wortes „Prüfung“ vor, die durch Anwendung der Tokenisierungsregel t_3 korrekt erkannt werden kann.

Die Regeln t_2 und t_3 vermeiden überdies insbesondere auch Wörter, die ausschließlich aus Sonderzeichen bestehen. Es lassen sich mit diesen Regeln jedoch nicht alle Probleme hinsichtlich der fehlerhaften Erkennung von Sonderzeichen vollständig lösen. Wenn ein Sonderzeichen von der Handschrifterkennung falsch erkannt und durch ein anderes Sonderzeichen ersetzt wird, kann dieser Fehler offenbar durch die

Anwendung der Regeln korrigiert werden. Dies trifft beispielsweise zu, wenn statt dem Zeichen „(“ das Zeichen „[“ erkannt wird. Falls allerdings statt „(“ fälschlicherweise der Großbuchstabe „C“ erkannt wird, ist dies durch die Tokenisierungsregeln nicht mehr korrigierbar. Solche Probleme müssen in einem nachgelagerten Schritt gesondert betrachtet werden. Dies wird im Detail in Abschnitt 3.7 diskutiert.

Zusammenfassend kommen somit die folgenden Tokenisierungsregeln zum Einsatz, um den in Abschnitt 3.6.1 diskutierten Problemen zu begegnen:

t_1 – Alle Buchstaben des Textes werden in Kleinbuchstaben umgewandelt. Danach wird der Text an Leerzeichen aufgetrennt. Das resultierende Tupel enthält die Wörter des Textes als Elemente in der Reihenfolge, in der sie im Text vorkommen. Darüber hinaus werden keine zusätzlichen Ersetzungen oder Modifikationen im Text durchgeführt.

Diese Regel kann formal als Abbildung wie folgt beschrieben werden:

$$t_1 : \mathcal{A}^* \rightarrow \mathcal{A}^{*^m} \text{ mit } t_1(s) := \text{Split}^+(\text{ToLower}(s), \beta).$$

t_2 – Alle Buchstaben des Textes werden in Kleinbuchstaben umgewandelt. Danach werden alle Leer- und Sonderzeichen aus dem Text entfernt. Der hierdurch entstehende Text wird nicht aufgetrennt, so dass ein Tupel mit nur einem einzigen Element entsteht.

Diese Regel kann formal als Abbildung wie folgt beschrieben werden:

$$t_2 : \mathcal{A}^* \rightarrow \mathcal{A}^{*^m} \text{ mit } t_2(s) := (\text{RemoveAll}(\text{ToLower}(s), \mathcal{S})).$$

t_3 – Alle Buchstaben des Textes werden in Kleinbuchstaben umgewandelt. Danach werden alle Sonderzeichen im Text durch Leerzeichen ersetzt, bevor der Text an den Leerzeichen aufgetrennt wird. Das resultierende Tupel enthält die Wörter des Textes als Elemente in der Reihenfolge, in der sie im Text vorkommen.

Diese Regel kann formal als Abbildung wie folgt beschrieben werden:

$$t_3 : \mathcal{A}^* \rightarrow \mathcal{A}^{*^m} \text{ mit } t_3(s) := \text{Split}^+(\text{ReplaceAll}(\text{ToLower}(s), \mathcal{S} \setminus \beta, \beta), \beta).$$

Abschließend ist zu bemerken, dass die Tokenisierungsregeln abgesehen von der Sonderbehandlung von Leerzeichen nicht weiter zwischen anderen verschiedenen Sonderzeichen unterscheiden. Der AugIR soll insbesondere die Kommunikation in heterogenen Stakeholder-Teams in frühen Phasen der Softwareentwicklung unterstützen. An den entsprechenden Diskussionen nehmen auch nicht-technische Stakeholder teil, weshalb die Gespräche in der Regel auf einem hohen Abstraktionsniveau geführt werden, ohne dabei zu viele technische Details zu betrachten. Die gemeinsam erstellten Skizzen enthalten deshalb für gewöhnlich keine technischen Details wie Klassen- oder Methodennamen. Aus diesem Grund ist es gerechtfertigt, nicht dediziert zwischen verschiedenen Symbolen zu unterscheiden, die häufig in technischen Diskussionen über Software vorkommen, wie zum Beispiel „.“, „::“ oder „#“.

3.7. Modifikation der Tokenisierungen

Im vorherigen Abschnitt wurden heuristische Regeln definiert, die erkannte Handschrifttexte in Tupel von Wörtern – sogenannte Tokenisierungen – zerlegen. Es wurde deutlich, warum mehrere verschiedene Regeln angewendet werden müssen, um bestimmten Erkennungsproblemen entgegenzuwirken.

Die Diskussion am Ende von Abschnitt 3.6.2 machte dabei jedoch bereits deutlich, dass die vorgestellten Regeln alleine nicht ausreichen, um alle potentiellen Schwierigkeiten zu beseitigen.

Im Folgenden wird deshalb ein Verfahren zur Modifikation der erzeugten Tokenisierungen präsentiert. Diese werden analysiert und gegebenenfalls angepasst, indem darin enthaltene Elemente durch hinreichend ähnliche Elemente ersetzt werden. Hierdurch kann die Korrelation potentiell zusammenhängender Elemente und damit die Erfassung textueller Beziehungen signifikant verbessert werden.

3.7.1. Vorüberlegungen und Definitionen

Sonderzeichen, die von der Handschrifterkennung fälschlicherweise durch ähnlich aussehende Buchstaben ersetzt wurden, können durch Anwendung der Tokenisierungsregeln nicht zuverlässig korrigiert werden. Weiterhin kann es gelegentlich passieren, dass die Handschrifterkennung auch gewöhnliche Buchstaben falsch erkennt. Dies hängt im Wesentlichen von der Form der gezeichneten Buchstaben in Kombination mit der Handschrift des jeweiligen Stakeholders ab. Insbesondere tritt dieses Problem häufig bei optisch ähnlichen Zeichen auf, wie beispielsweise bei `t` und `f` oder `u` und `v`. Auch solche Fehler können durch die Tokenisierungen nicht behoben werden.

Darüber hinaus kann die Schreibweise bestimmter Wörter in verschiedenen Skizzen voneinander abweichen. Zum einen kann dies unbeabsichtigt passieren, indem Stakeholder ein Wort falsch schreiben oder versehentlich ein leicht abgewandeltes Wort verwenden. Zum anderen kann dies auch absichtlich geschehen, indem Stakeholder bewusst verschiedene Formen des gleichen Wortes an unterschiedlichen Stellen benutzen. Beispielsweise könnte das Objekt „Kunde“ im Singular auf dem Object Canvas vorkommen, während auf dem Process Canvas eine Aktion „Kunden informieren“ existiert, die das gleiche Wort im Plural enthält. Offensichtlich bezieht sich diese Aktion auf das vorgenannte Objekt des Object Canvases, weshalb ein eindeutiger Zusammenhang besteht. Eine Tokenisierung kann jedoch unterschiedliche Wortformen in der Regel nicht angleichen oder korrigieren. Deshalb ergibt sich im vorliegenden Fall unter Anwendung jeder Tokenisierungsregel jeweils eine Kosinus-Ähnlichkeit von 0, wie man leicht nachrechnet. Es kann in diesem Fall also kein Zusammenhang zwischen den betreffenden Elementen identifiziert werden, obwohl ein solcher vorliegt.

Ein ähnliches Problem kann ebenfalls auftreten, wenn Verben in verschiedenen Modi verwendet werden. Hierzu betrachtet man beispielsweise den unterschiedlichen Modus des Verbs „starten“ in den Texten „Softwareentwicklung starten“ und „Starte Softwareentwicklung“. Genau wie im vorherigen Fall kann auch hier eine Tokenisierung der Texte die Wortformen nicht angleichen. Die Anwendung der ersten Tokenisierungsregel t_1 führt deshalb zu einer Kosinus-Ähnlichkeit von 0,5, während die

Regeln t_2 und t_3 sogar in einer Kosinus-Ähnlichkeit von 0 resultieren.

Um die vorgenannten Probleme zu lösen, werden die erzeugten Tokenisierungen in einem zusätzlichen Zwischenschritt analysiert und falls nötig modifiziert, bevor schließlich mit Hilfe des Vector Space Models eine Erfassung der Trace Links stattfindet.

Die Grundidee hierbei ist Folgende: Für jedes Wort einer Tokenisierung eines Textes aus einer Skizze S_1 wird geprüft, ob dieses in einer anderen betrachteten Skizze S_2 vorkommt. Falls dies der Fall ist, wird das betreffende Element im Tupel nicht verändert. Andernfalls wird das Element durch ein ausreichend ähnliches Wort aus Skizze S_2 ersetzt, sofern die Ähnlichkeit zu diesem hoch genug ist. Hierdurch kann die Korrelation zwischen potentiell zusammenhängenden Elementen unterschiedlicher Skizzen erhöht und damit die Qualität des nachfolgenden Traceability-Algorithmus verbessert werden.

Zur Bestimmung der Ähnlichkeit zwischen zwei Wörtern wird das in Abschnitt 3.2.7 eingeführte Ähnlichkeitsmaß σ verwendet. Wenn für zwei Wörter $w_1, w_2 \in \mathcal{A}^*$ und einen Schwellwert $\mu \in [0,1] \subset \mathbb{R}$ gilt, dass $\sigma(w_1, w_2) \geq \mu$ ist, dann liegt eine hinreichend große Ähnlichkeit zwischen beiden Wörtern vor und w_1 wird wie zuvor beschrieben in der Tokenisierung durch w_2 ersetzt. Ein sinnvoller Wert für den Schwellwert μ wird später im nachfolgenden Abschnitt ermittelt.

Für das vorgenannte Beispiel mit den Dokumenten „Kunde“ und „Kunden informieren“ ergibt sich beispielsweise für die Wörter „kunde“ und „kunden“ eine Ähnlichkeit von

$$\sigma(\text{„kunde“}, \text{„kunden“}) = 1 - \frac{\text{lev}(\text{„kunde“}, \text{„kunden“})}{\max(|\text{„kunde“}|, |\text{„kunden“}|)} = 1 - \frac{1}{6} = 0{,}833.$$

Unter der Annahme, dass das Wort „kunde“ in keinem anderen Dokument der zweiten Skizze vorkommt, würde es für einen Schwellwert $\mu \leq 0{,}833$ vor der Berechnung der Kosinus-Ähnlichkeit in der Tokenisierung durch das Wort „kunden“ ersetzt. Dies führt zur Wortmenge

$$W := \{\text{„kunden“}, \text{„informieren“}\}.$$

Die entsprechenden Häufigkeitsvektoren ergeben sich in diesem Fall zu $q = (1, 0)$ und $d = (1, 1)$, was in einer angemessen hohen Kosinus-Ähnlichkeit von

$$\text{CosSim}((1,0),(1,1)) = \frac{(1,0)\cdot(1,1)}{|(1,0)|\cdot|(1,1)|} = \frac{1}{\sqrt{2}} = 0{,}7071$$

resultiert.

Entsprechend ergibt sich im zweiten Beispiel für die Wörter „starten“ und „starte“ in den Dokumenten „Softwareentwicklung starten“ und „Starte Softwareentwicklung“ eine Ähnlichkeit von

$$\sigma(\text{„starten“}, \text{„starte“}) = 1 - \frac{\text{lev}(\text{„starten“}, \text{„starte“})}{\max(|\text{„starten“}|, |\text{„starte“}|)} = 1 - \frac{1}{7} = 0{,}8571.$$

Wieder unter der Annahme, dass das Wort „starten“ in keinem anderen Dokument der zweiten Skizze vorkommt, würde für einen Schwellwert $\mu \leq 0{,}8571$ vor der Berechnung der Kosinus-Ähnlichkeit das Wort „starten“ in der Tokenisierung durch das Wort „starte“ ersetzt. Hiermit ergibt sich die Wortmenge W zu

$$W := \{\text{„softwareentwicklung“}, \text{„starte“}\}.$$

Die entsprechenden Häufigkeitsvektoren sind $q = d = (1, 1)$, wodurch sich sogar eine Kosinus-Ähnlichkeit von

$$\text{CosSim}((1,1),(1,1)) = \frac{(1,1)\cdot(1,1)}{|(1,1)|\cdot|(1,1)|} = \frac{2}{\sqrt{2}\cdot\sqrt{2}} = 1$$

ergibt. Es kann für diese Elemente also ein perfekter Treffer identifiziert werden.

Formal lässt sich die zuvor beschriebene Ersetzung als Abbildung

$$\text{ReplaceSim} : \mathcal{A}^{*^m} \times \mathcal{P}(\mathcal{A}^*) \times \mathbb{R} \to \mathcal{A}^{*^m}$$

wie folgt definieren:

Es sei $(w_1, \ldots, w_m) \in \mathcal{A}^{*^m}$ eine Tokenisierung mit m Zeichenketten beliebiger Länge, $W \in \mathcal{P}(\mathcal{A}^*)$ eine Menge von Zeichenketten über $\mathcal{A}$ sowie $\mu \in \mathbb{R}$ ein konstanter

Schwellwert. Dann ist $\text{ReplaceSim}((w_1, \ldots, w_m), W, \mu) := (w'_1, \ldots, w'_m)$ mit

$$w'_i := \begin{cases} w_i, & \text{falls } w_i \in W, \\ w' \in W, & \text{falls } w_i \notin W \ \wedge \ \sigma(w_i, w') = \max\{\sigma(w_i, w) \mid w \in W\} \geq \mu, \\ w_i, & \text{sonst} \end{cases}$$

für $i = 1, \ldots, m$. Das i-te Element der Tokenisierung wird demnach unter dieser Abbildung jeweils durch ein Wort $w' \in W$ ersetzt, wenn w_i nicht in W vorkommt, wenn w' von allen Elementen aus W zu w_i die höchste Ähnlichkeit aufweist und wenn diese Ähnlichkeit über einem gegebenen Schwellwert μ liegt. Welcher konkrete Schwellwert μ hierbei die besten Ergebnisse liefert, wird im Rahmen eines Experiments im nachfolgenden Abschnitt untersucht.

Wie die vorangegangen Beispiele illustrieren, können bestimmte Probleme bei der Handschrifterkennung, die sich nicht durch die Wahl geeigneter Tokenisierungsregeln lösen lassen, also dadurch gemindert werden, dass vor der Berechnung der Kosinus-Ähnlichkeit in einem weiteren Zwischenschritt die erzeugten Tokenisierungen modifiziert werden. Hierbei werden Wörter, die in einer betrachteten Skizze nicht vorkommen, durch hinreichend ähnliche Wörter dieser Skizze ersetzt. Die vorherigen Beispiele zeigen, dass hierdurch die Kosinus-Ähnlichkeit für zusammengehörige Elemente signifikant erhöht werden kann. Obwohl sich durch diesen Zwischenschritt auch falsche Zuordnungen ergeben können, wird man als Ergebnis des nachfolgenden Experiments sehen, dass sich hierdurch viele potentielle Erkennungsfehler eliminieren lassen.

3.7.2. Ermittlung eines optimalen Schwellwerts

In diesem Abschnitt wird untersucht, welcher Schwellwert μ bei der Ähnlichkeitsbetrachtung und der Ersetzung von Elementen in den erzeugten Tokenisierungen die besten Ergebnisse liefert.

Versuchsaufbau

Um potentiell falsch erkannte Wörter in den Tokenisierungen durch hinreichend ähnliche Wörter aus anderen Skizzen ersetzen zu können, muss ein geeigneter Schwellwert μ als Grenzwert für die Ähnlichkeitsbetrachtung identifiziert werden. Die Wahl eines optimalen Schwellwerts ist wichtig, weil ein zu niedrig gewählter Wert zu falschen Ersetzungen durch unpassende Wörter führen kann, während bei Verwendung eines zu hohen Wertes unter Umständen nicht alle falsch erkannten Wörter ersetzt werden.

Um einen passenden Schwellwert zu ermitteln, wurde ein Experiment durchgeführt. Hierzu wurden die Inhalte der in Abschnitt 3.3 beschriebenen digital rekonstruierten Interaction-Room-Projekte analysiert. Jedem Wort, das vom Handschrifterkenner falsch erkannt wurde, wurde jeweils das ähnlichste Wort aus der zugehörigen originalen analogen Skizze zugeordnet. Zur Berechnung der Ähnlichkeit kam hierbei das in Abschnitt 3.2.7 eingeführte Ähnlichkeitsmaß σ zum Einsatz. Für jede Zuordnung wurde geprüft, wie hoch die Ähnlichkeit zwischen den beiden zugeordneten Wörtern war und ob auf diese Weise für ein falsch erkanntes Wort tatsächlich das richtige Wort aus der Ursprungsskizze ermittelt werden konnte. Auf diese Weise lässt sich untersuchen, für welche falsch erkannten Wörter sich passende korrekte Wörter finden lassen und welche Schwellwerte hierzu verwendet werden müssen.

Um die falsch erkannten handgeschriebenen Wörter zu klassifizieren, werden im Folgenden drei Definitionen verwendet:

- Ein falsch erkanntes handgeschriebenes Wort wird als *potentiell zuordenbar* bezeichnet, wenn es seinem korrekten Gegenstück zugeordnet werden konnte und zu diesem eine Ähnlichkeit > 0 aufweist.
- Ein falsch erkanntes handgeschriebenes Wort wird als *zuordenbar* (*für den Schwellwert* $\mu \in]0{,}1[$) bezeichnet, wenn es seinem korrekten Gegenstück zugeordnet werden konnte und zu diesem eine Ähnlichkeit $\geq \mu$ aufweist.

Tabelle 3.16.: Analyse von 16 digital rekonstruierten Projekten hinsichtlich potentiell zuordenbarer Wörter.

Projekt	Wörter	Falsch erkannt	Potentiell zuordenbar	% Wörter potentiell zuordenbar
P_1	242	22	17	77,27
P_2	88	3	0	0
P_3	216	9	5	55,56
P_4	60	5	4	80
P_5	207	21	19	90,48
P_6	187	19	14	73,68
P_7	226	27	21	77,78
P_8	304	26	18	69,23
P_9	218	30	15	50
P_{10}	192	25	13	52
P_{11}	154	7	6	85,71
P_{12}	322	12	9	75
P_{13}	347	23	18	78,26
P_{14}	332	55	36	65,46
P_{15}	191	13	10	76,92
P_{16}	830	123	69	56,1
Gesamt	4 116	420	274	65,24

- Ein falsch erkanntes handgeschriebenes Wort wird als *nicht zuordenbar* bezeichnet, wenn es nicht potentiell zuordenbar ist, d. h. wenn es entweder einem falschen Wort oder gar keinem Wort zugeordnet wurde.

Ergebnisse

Tabelle 3.16 fasst die Resultate der zuvor beschriebenen Analyse für alle 16 Projekte hinsichtlich potentiell zuordenbarer Wörter zusammen. Die erste Spalte zeigt die Kürzel der Projekte. Die zweite Spalte führt die Gesamtzahl aller handgeschriebenen Wörter in jedem Projekt auf. Diese ergibt sich als Summe aller Wörter über alle Skizzen des betrachteten Projekts. Die dritte Spalte zeigt die Anzahl der Wörter, die in jedem Projekt von der Handschrifterkennung falsch erkannt wurden. In der

vierten Spalte ist angegeben, wie viele dieser falsch erkannten Wörter potentiell zuordenbar sind. Die letzte Spalte zeigt diesen Wert als prozentualen Anteil.

Insgesamt enthalten die 16 analysierten Projekte 4 116 Wörter. Die Handschrifterkennung konnte hiervon 420 Wörter nicht korrekt erkennen. Die entspricht einer Fehlerrate von 10,2 %.

Bei Betrachtung der Tabelle fällt sofort auf, dass der Prozentanteil der potentiell zuordenbaren Wörter für fast jedes Projekt deutlich über 50 % liegt. Lediglich einige Projekte wie beispielsweise P_2 und P_9 enthalten jedoch domänenspezifische Begriffe und kryptische Abkürzungen, die von der Handschrifterkennung nur schwer fehlerfrei erkannt werden können. Weil die erkannten Wörter deutlich von ihrem fehlerfreien Gegenstück abweichen und in keinem Wörterbuch vorkommen, kann ihnen aufgrund zu geringer Ähnlichkeiten häufig kein korrektes Wort zugeordnet werden.

Durch Verwendung des zuvor präsentierten Verfahrens konnten insgesamt 274 von 420 falsch erkannten Wörtern korrigiert werden. Dies entspricht 64,24 % aller falsch erkannten Wörter. 146 Wörter konnten nicht korrekt zugeordnet werden, was 34,76 % aller falsch erkannten Wörter entspricht.

Als Zwischenfazit lässt sich somit bereits festhalten, dass bei der Wahl des richtigen Schwellwerts eine große Anzahl falsch erkannter Wörter potentiell korrigierbar ist, sofern das entsprechende Wort in seiner fehlerfreien Form in der betrachteten Skizze vorkommt.

Im nächsten Schritt soll deshalb untersucht werden, welcher Grenzwert μ sich hierfür am besten eignet. Tabelle 3.17 zeigt, wie viele falsch erkannte Wörter bei Verwendung eines Schwellwerts im Bereich zwischen 0,9 und 0,6 jeweils korrekt zugeordnet werden können.

Spalte 1 listet erneut die Projektkürzel auf. In den Spalten 2 bis 6 sind die Anzahl der korrekten Zuordnungen sowie die Gesamtzahl aller Zuordnungen für den jeweiligen Ähnlichkeitsschwellwert aufgeführt.

Spalte 2 zeigt, dass insgesamt 137 falsch erkannte Wörter zuordenbar für den Schwellwert $\mu := 0{,}9$ sind. Diese Wörter weisen demnach eine Ähnlichkeit $\geq 0{,}9$ zu ihrem

Tabelle 3.17.: Anzahl der korrekt und insgesamt zugeordneten Wörter bei Verwendung eines Ähnlichkeitsschwellwerts μ zwischen 0,9 und 0,6.

	Korrekte Zuordnungen / Gesamte Zuordnungen für $\mu := \ldots$				
Projekt	0,9	0,8	0,75	0,7	0,6
P_1	12 / 12	14 / 14	14 / 14	14 / 14	17 / 19
P_2	0 / 0	0 / 0	0 / 0	0 / 1	0 / 1
P_3	4 / 4	5 / 5	5 / 5	5 / 5	5 / 5
P_4	2 / 2	4 / 4	4 / 4	4 / 4	4 / 4
P_5	14 / 14	16 / 16	18 / 18	19 / 20	19 / 20
P_6	4 / 4	11 / 11	12 / 12	13 / 13	14 / 14
P_7	9 / 9	13 / 13	16 / 16	16 / 16	20 / 20
P_8	7 / 7	15 / 15	15 / 15	15 / 15	18 / 19
P_9	5 / 5	13 / 13	14 / 14	15 / 15	15 / 18
P_{10}	9 / 9	10 / 10	11 / 11	11 / 12	13 / 17
P_{11}	4 / 4	5 / 5	6 / 6	6 / 6	6 / 7
P_{12}	7 / 7	9 / 9	9 / 10	9 / 10	9 / 10
P_{13}	11 / 11	16 / 16	17 / 17	18 / 18	18 / 19
P_{14}	15 / 15	31 / 31	32 / 33	32 / 36	35 / 44
P_{15}	6 / 6	8 / 8	9 / 9	9 / 9	9 / 10
P_{16}	28 / 28	47 / 47	58 / 58	58 / 60	68 / 74
Gesamt	137 / 137	217 / 217	240 / 242	244 / 254	270 / 301
% korrekt	100	100	99,17	96,06	89,7

zugehörigen fehlerfreien Wort aus der analogen Originalskizze auf und können durch Anwendung des zuvor beschriebenen Verfahrens in den Tokenisierungen korrigiert werden, sofern das fehlerfreie Wort an anderer Stelle in einer Skizze existiert. Für $\mu = 0{,}9$ lassen sich somit bereits insgesamt 32,62 % aller insgesamt falsch erkannten Wörter und 50 % aller potentiell zuordenbaren Wörter korrigieren. Weil alle Zuweisungen korrekt waren und kein einziges Wort falsch zugeordnet wurde, liegt für $\mu = 0{,}9$ eine Fehlerrate von 0 % vor.

Spalte 3 zeigt, dass für $\mu := 0{,}8$ insgesamt 217 falsch erkannte Wörter zuordenbar sind. Dies sind 80 Wörter mehr als bei Verwendung des Schwellwerts 0,9. Bei Absenkung des Grenzwerts können somit zusätzliche Wörter korrigiert werden, was zu erwarten war. Auch für $\mu = 0{,}8$ sind alle Zuordnungen korrekt und die Fehlerrate beträgt weiterhin 0 %. Es können hierbei also 51,67 % aller insgesamt falsch erkannten Wörter und 79,2 % aller potentiell zuordenbaren Wörter korrigiert werden.

Aus Spalte 4 ist ersichtlich, dass bei Verwendung eines Schwellwerts von $\mu := 0{,}75$ insgesamt 240 falsch erkannte Wörter zuordenbar sind. Darüber hinaus werden hierbei jedoch auch 2 Wörter falsch zugeordnet, weshalb die Fehlerrate marginal auf 0,83 % ansteigt. Für einen Grenzwert von 0,75 können demnach 57,14 % aller insgesamt falsch erkannten Wörter und 87,59 % aller potentiell zuordenbaren Wörter korrigiert werden.

Spalte 5 zeigt, dass bei erneuter Absenkung des Schwellwerts auf $\mu := 0{,}7$ insgesamt 244 falsch erkannte Wörter zuordenbar sind. 10 Wörter werden jedoch bei Verwendung dieses Schwellwerts falsch zugeordnet, was die Fehlerrate auf 3,94 % erhöht. Gegenüber dem Grenzwert 0,75 werden zwar 4 zusätzliche Wörter korrigiert, es kommen jedoch auch 8 zusätzliche falsche Zuordnungen hinzu. Für $\mu = 0{,}7$ können somit 58,1 % aller insgesamt falsch erkannten Wörter und 89,05 % aller potentiell zuordenbaren Wörter korrigiert werden.

Spalte 6 illustriert, was bei einer weiteren Verringerung des Schwellwerts geschieht: Für $\mu := 0{,}6$ steigt die Anzahl der zuordenbaren Wörter auf 270. Allerdings werden 31 Wörter falsch zugeordnet, wodurch sich die Fehlerrate deutlich auf 10,3 % erhöht.

Tabelle 3.18.: Anzahl der korrekt und insgesamt zugeordneten Wörter bei Verwendung eines Ähnlichkeitsschwellwerts μ zwischen 0,5 und 0,1.

	Korrekte Zuordnungen / Gesamte Zuordnungen für $\mu := \ldots$				
Projekt	0,5	0,4	0,3	0,2	0,1
P_1	17 / 19	17 / 20	17 / 21	17 / 21	17 / 21
P_2	0 / 2	0 / 2	0 / 3	0 / 3	0 / 3
P_3	5 / 6	5 / 6	5 / 7	5 / 8	5 / 8
P_4	4 / 4	4 / 4	4 / 4	4 / 4	4 / 4
P_5	19 / 20	19 / 20	19 / 20	19 / 21	19 / 21
P_6	14 / 17	14 / 19	14 / 19	14 / 19	14 / 19
P_7	20 / 21	20 / 22	21 / 23	21 / 23	21 / 23
P_8	18 / 22	18 / 23	18 / 25	18 / 25	18 / 25
P_9	15 / 21	15 / 22	15 / 27	15 / 28	15 / 29
P_{10}	13 / 22	13 / 23	13 / 25	13 / 25	13 / 25
P_{11}	6 / 7	6 / 7	6 / 7	6 / 7	6 / 7
P_{12}	9 / 11	9 / 11	9 / 11	9 / 11	9 / 11
P_{13}	18 / 21	18 / 22	18 / 22	18 / 22	18 / 22
P_{14}	36 / 47	36 / 49	36 / 53	36 / 54	36 / 54
P_{15}	10 / 12	10 / 12	10 / 12	10 / 12	10 / 12
P_{16}	69 / 93	69 / 103	69 / 109	69 / 121	69 / 122
Gesamt	273 / 345	273 / 365	274 / 388	274 / 404	274 / 406
% korrekt	79,13	74,79	70,62	67,82	67,49

Erwartungsgemäß steigt die Fehlerrate weiter, je niedriger der Schwellwert gewählt wird. Dies wird durch Tabelle 3.18 illustriert, welche die Anzahl der korrekt und insgesamt zugeordneten Wörter bei Verwendung eines Ähnlichkeitsschwellwerts zwischen 0,5 und 0,1 zeigt.

Durch Absenkung des Schwellwerts auf 0,5 können lediglich 3 zusätzliche Wörter korrekt zugeordnet werden, wobei jedoch 72 falsche Zuweisungen zu beobachten sind. Dies verdoppelt in etwa die Fehlerrate und erhöht sie auf 20,87 %.

Durch weitere Verringerungen des Schwellwerts kann insgesamt lediglich ein zusätzliches Wort korrekt zugeordnet werden, während gleichzeitig die Anzahl der falschen Zuweisungen und damit die Fehlerrate drastisch ansteigen.

Insgesamt wird deshalb im Folgenden $\mu := 0{,}75$ als Ähnlichkeitsschwellwert gewählt, weil hierdurch 57,14 % aller insgesamt falsch erkannten Wörter und 87,59 % aller potentiell zuordenbaren Wörter korrigiert werden können und die Fehlerrate lediglich bei 0,83 % liegt. Eine Absenkung des Schwellwerts auf 0,7 erscheint nicht sinnvoll, weil hierdurch nur 4 zusätzliche Wörter korrigiert werden aber doppelt so viele falsche Zuordnungen hinzukommen.

Diskussion und Gefahren für die Gültigkeit

Durch Analyse der potentiell zuordenbaren Wörter wurde deutlich, dass durch die Wahl eines geeigneten Schwellwerts zahlreiche falsch erkannte Wörter korrigiert werden können, sofern sie in richtiger Schreibweise an anderer Stelle vorkommen. Ein Schwellwert von $\mu := 0{,}75$ erzielte hierbei hinsichtlich der rekonstruierten Interaction-Room-Projekte die besten Ergebnisse.

Die ermittelten Zahlen hängen dabei jedoch stark von der Anzahl und Schwere der Handschrifterkennungsfehler ab. Diese resultieren wiederum unmittelbar aus der Handschrift der Personen, welche die Skizzen digital rekonstruiert haben. Obwohl diese von fünf verschiedenen Personen gezeichnet wurden, um möglichst viele unterschiedliche Handschriften zu berücksichtigen, existieren sicherlich zahlreiche weitere Handschrifttypen, die in diesem Experiment nicht berücksichtigt werden konnten.

Darüber hinaus wurde der beste Ähnlichkeitsschwellwert durch einen Vergleich der rekonstruierten Skizzen mit ihren originalen Gegenstücken ermittelt. In der Praxis wird der Algorithmus jedoch selten zwei verschiedene Versionen der gleichen Skizze sondern stattdessen eher verschiedene Skizzen mit ähnlichen oder gemeinsamen textuellen Inhalten miteinander vergleichen. Dennoch kann der zuvor beschriebene Ansatz die erzeugten Tokenisierungen hinsichtlich der Trace-Link-Erfassung sinnvoll korrigieren, sofern die falsch erkannten Wörter in korrekter Schreibweise an anderer Stelle vorkommen.

Weil die analysierten Projekte aus unterschiedlichen Domänen stammen und die Inhalte sehr heterogen sind, kann angenommen werden, dass sich die Resultate zu-

mindest auf zukünftige Interaction-Room-Projekte in deutscher Sprache übertragen lassen.

3.8. Erfassung von Trace Links

Mit Hilfe der in den vorangegangenen Abschnitten gewonnenen Erkenntnisse kann nun ein Algorithmus angegeben werden, der textuell ähnliche Elemente verschiedener Skizzen durch Anwendung des Vector Space Models miteinander verknüpft. Bevor dieser mit mathematischen Mitteln präzise beschrieben wird, soll der Ablauf zur Erfassung von Trace Links zwischen zwei Skizzen S_1 und S_2 kurz informell skizziert werden. Er besteht im Wesentlichen aus den folgenden drei Schritten:

Zunächst werden auf den Text jedes Elements e aus S_1 die drei Tokenisierungsregeln t_1, t_2 und t_3 angewendet. Die erzeugten Tokenisierungen werden falls nötig wie in Abschnitt 3.7 erläutert modifiziert, indem Wörter in diesen Tupeln durch hinreichend ähnliche Wörter aus der Skizze S_2 ersetzt werden, um die Korrelation zwischen potentiell zusammenhängenden Elementen zu erhöhen.

Danach wird für jede der drei modifizierten Tokenisierungen jeweils ein Häufigkeitsvektor erzeugt, der mit Hilfe der in Abschnitt 3.2.8 eingeführten Kosinus-Ähnlichkeit mit dem Häufigkeitsvektor der entsprechenden Tokenisierung jedes Elements h aus Skizze S_2 verglichen wird. Hierdurch kann die textuelle Ähnlichkeit zwischen e und h bestimmt werden.

Abschließend werden die zuvor errechneten Kosinus-Ähnlichkeiten mit einem vorgegebenen Schwellwert ν verglichen. Ist mindestens eine davon größer oder gleich diesem Schwellwert, so liegt eine hinreichend große Ähnlichkeit für mindestens eine Tokenisierung vor und es wird ein Trace Link zwischen e und h erfasst.

Im folgenden Abschnitt werden zunächst einige Begrifflichkeiten definiert, bevor danach der zuvor skizzierte Ablauf formal präzise angegeben wird.

3.8.1. Definitionen

Seien $S_1, \ldots, S_n$ die Skizzen in einem AugIR-Projekt. Dann enthält jede Skizze S_j eine beliebige Anzahl handgezeichneter Elemente, wie beispielsweise Aktionen in einem Process Canvas oder Objekte in einem Object Canvas. Diese Elemente werden gebildet, indem logisch zusammengehörige Linienzüge wie in Abschnitt 3.5 beschrieben automatisch zu Linienzuggruppen zusammengefasst werden. Jedes Element besteht somit in der Regel sowohl aus Linienzügen, die Text repräsentieren, als auch aus Linienzügen, die geometrische Formen beschreiben. Die Menge aller in einer Skizze S_i enthaltenen Elemente sei mit $E(S_i)$ bezeichnet.

Jedes Element $e \in E(S_i)$ sei definiert als ein Tripel $e := (d_e, s_e, A_e)$. Hierbei bezeichne d_e den Text, der von der Handschrifterkennung aus den Text-Linienzügen des Elements erkannt wurde. Dieser wird im Kontext des Information Retrievals häufig auch als *Dokument* bezeichnet. Er repräsentiert den unmodifizierten Text eines Elements vor der Anwendung der Tokenisierungsregeln. Ein Dokument besteht somit aus beliebigen Zeichen und kann insbesondere auch Leer- und Sonderzeichen enthalten. Weiterhin beschreibe s_e die geometrische Freihandform des Elements, also die Menge aller Linienzüge, die als Form-Linienzüge klassifiziert wurden. A_e sei die Multimenge der auf diesem Element platzierten Annotationen.

Eine Aktion „Vertrag unterzeichnen“ in einem Process Canvas kann dann beispielsweise durch ein Element e dargestellt werden, das von einem rechteckigen Kasten s_e umgeben ist und dessen Dokument d_e den Text „Vertrag unterzeichnen“ enthält. Die Menge A_e ist leer, wenn keine Annotationen auf dem entsprechenden Element platziert wurden.

Es ist zu beachten, dass Skizzen sowohl Text ohne zugehörige Freihandformen (z. B. Anmerkungen oder textuelle Notizen) als auch Freihandformen ohne Text (z. B. bestimmte Symbole wie Pfeile und Rauten) enthalten können. In Abschnitt 3.4 wurde beschrieben, wie durch Anwendung heuristischer Regeln eine Unterscheidung zwischen Text- und Form-Linienzügen vorgenommen werden kann. Formen ohne Text, d. h. Elemente mit einem leeren Dokument d_e, werden im Folgenden ignoriert, weil nur zwischen textuellen Elementen sinnvolle Trace Links erfasst werden können.

Wie in Abschnitt 3.6 erläutert, werden im Rahmen der Tokenisierung aus jedem Dokument d_e drei Tupel erzeugt, indem auf dieses die Tokenisierungsregeln $t_1, \ldots, t_3$ angewendet werden. Für $i = 1, \ldots, 3$ sei somit $t_i(d_e) := (t_{i_1}, \ldots, t_{i_m})$, wobei t_{i_k} für $k = 1, \ldots, m$ jeweils die Wörter bezeichne, die durch die Anwendung der entsprechenden Tokenisierungsregel t_i auf das Dokument d_e entstehen.

Schließlich werden für jede Skizze S_j und jede Tokenisierung t_i mit $1 \leq i \leq 3$ noch die beiden folgenden Mengen definiert:

$T_i(S_j) := \{t_i(d_e) \mid e \in E(S_j)\}$ bezeichne die Menge aller Tokenisierungen, die aus den Dokumenten aller Elemente einer Skizze S_j durch Anwendung der Tokenisierungsregel t_i entstehen.

$W_i(S_j) := \{w \mid \exists\ u \in T_i(S_j) \text{ mit } w \in u\}$ bezeichne die Menge aller Wörter, die in mindestens einer von t_i erzeugten Tokenisierung eines Dokuments in Skizze S_j vorkommen.

3.8.2. Algorithmus zur Erfassung von Trace Links

Wenn ein Element e in einer Skizze S gezeichnet oder verändert wird, erstellt oder aktualisiert dies das zugehörige Tripel $e = (d_e, s_e, A_e)$. Für jede Skizze S_j mit $S_j \neq S$ werden daraufhin Trace Links zwischen e und Elementen aus S_j wie folgt in drei Schritten erfasst:

Schritt 1: Erstellung der modifizierten Tokenisierungen

Für jede Tokenisierung $t_i(d_e) := (t_{i_1}, \ldots, t_{i_m})$ mit $1 \leq i \leq 3$ wird eine modifizierte Tokenisierung $t'_i(d_e) := (t'_{i_1}, \ldots, t'_{i_m})$ durch Anwendung der in Abschnitt 3.7 definierten Abbildung ReplaceSim : $\mathcal{A}^{*^m} \times \mathcal{P}(\mathcal{A}^*) \times \mathbb{R} \to \mathcal{A}^{*^m}$ erzeugt mit

$$t'_i(d_e) := \text{ReplaceSim}(t_i(d_e),\ W_i(S_j),\ 0{,}75).$$

Hierbei wird der in Abschnitt 3.7.2 ermittelte Schwellwert $\mu := 0{,}75$ verwendet. Jedes Element t'_{i_k} des hierdurch erzeugten Wort-Tupels $t'_i(d_e)$ stimmt somit entweder

- mit dem entsprechenden Wort $t_{i_k} \in t_i(d_e)$ überein, falls t_{i_k} in der von t_i erzeugten Tokenisierung eines Dokuments in Skizze S_j vorkommt, oder
- mit demjenigen Wort aus der von t_i erzeugten Tokenisierung eines Dokuments aus S_j überein, das hinreichend ähnlich zu t_{i_k} ist, oder
- mit dem entsprechenden Wort t_{i_k} überein, falls kein hinreichend ähnliches Wort in der von t_i erzeugten Tokenisierung eines Dokuments in Skizze S_j existiert.

Schritt 2: Bestimmung der Ähnlichkeit

Im zweiten Schritt wird die Ähnlichkeit zwischen dem Element e aus Skizze S und allen Elementen $h \in E(S_j)$ bestimmt.

Hierzu muss für jede modifizierte Tokenisierung $t'_i(d_e)$ mit $1 \leq i \leq 3$ ein numerischer Häufigkeitsvektor bestimmt und unter Anwendung der Kosinus-Ähnlichkeit mit dem entsprechenden Häufigkeitsvektor der Tokenisierung $t_i(d_h)$ verglichen werden.

Die Ähnlichkeit zwischen beiden Elementen bestimmt sich dann für die i-te Tokenisierung jeweils durch $\mathrm{CosSim}_{t'_i(d_e),t_i(d_h)}$. Sie kann als Indiz dafür betrachtet werden, wie ähnlich die Elemente e und h hinsichtlich ihrer textuellen Inhalte sind.

Schritt 3: Erfassung von Trace Links

Im letzten Schritt werden Trace Links zwischen e und den Elementen $h \in E(S_j)$ erfasst, falls die zuvor berechneten Kosinus-Ähnlichkeiten zwischen ihnen für mindestens eine Tokenisierung $1 \leq i \leq 3$ über einem bestimmten Schwellwert ν liegen. Hierzu werden zwei Fälle unterschieden:

1. Falls $\text{CosSim}_{t'_i(d_e),t_i(d_h)} \geq \nu$, wird ein Trace Link zwischen e und h erstellt.

   Falls bereits ein Trace Link zwischen diesen Elementen existiert und falls die neu errechnete Kosinus-Ähnlichkeit größer ist als der zuvor ermittelte Wert für den bereits existierenden Trace Link, so wird dieser entsprechend aktualisiert. Für jeden Trace Link wird die zugehörige Kosinus-Ähnlichkeit gespeichert. Sie gibt die Stärke des Trace Links an, d. h. wie eng die zugehörigen Elemente textlich zusammenhängen.

2. Falls $0 < \text{CosSim}_{t'_i(d_e),t_i(d_h)} < \nu$ ist und falls noch kein Trace Link zwischen e und h existiert, werden die beiden folgenden Bedingungen geprüft:

   a) Falls $\forall\, t'_{i_k} \in t'_i(d_e) : t'_{i_k} \in t_i(d_h)$, d. h. falls alle Elemente der modifizierten Tokenisierung $t'_i(d_e)$ im Tupel $t_i(d_h)$ vorkommen, wird ein Trace Link zwischen e und h erstellt und dessen Kosinus-Ähnlichkeit auf ν gesetzt.

   b) Falls $\forall\, t_{i_k} \in t_i(d_h) : t_{i_k} \in t'_i(d_e)$, d. h. falls alle Elemente der Tokenisierung $t_i(d_h)$ im Tupel $t'_i(d_e)$ vorkommen, wird ein Trace Link zwischen h und e erstellt und dessen Kosinus-Ähnlichkeit auf ν gesetzt.

   Die beiden letzten Bedingungen sind notwendig, um Beziehungen zwischen zusammenhängenden Elementen zu behandeln, deren Texte eine deutlich unterschiedliche Anzahl von Wörtern aufweisen. Texte auf bestimmten Skizzen wie Object und Integration Canvases bestehen häufig nur aus wenigen Wörtern. Im Gegensatz dazu enthalten Elemente auf anderen Skizzen wie beispielsweise dem Feature oder Process Canvas in der Regel deutlich mehr Wörter. Dies kann unter Umständen zu übersehenen Trace Links führen, weil die Kosinus-Ähnlichkeit zwischen zwei Dokumenten gering ist, wenn zwar ein Dokument alle Wörter eines anderen enthält, aber zusätzlich weitere Wörter umfasst, die in dem anderen Dokument nicht vorkommen. Beispielsweise existiert offensichtlich ein Zusammenhang zwischen den Elementen „Projekt“ und „Initialisiere Projekt Management System“. Obwohl diese Elemente einen starken inhaltlichen Zusammenhang aufweisen, ist ihre entsprechende Kosinus-Ähnlichkeit mit

$$\text{CosSim}((0,1,0,0),(1,1,1,1)) = \frac{1}{1 \cdot \sqrt{4}} = 0{,}5$$

nur verhältnismäßig gering, weil sie lediglich in einem Wort übereinstimmen und der Text des zweiten Elements drei Wörter enthält, die im Text des ersten Elements nicht vorkommen. Deshalb wird in bestimmten Fällen auch bei niedriger Kosinus-Ähnlichkeit ein Trace Link erstellt, sofern alle Wörter des Textes eines Elements in dem Text des anderen Elements enthalten sind.

Es ist zu bemerken, dass ein exakter Wert für den Schwellwert ν noch festgelegt werden muss. Dies erfolgt in Kapitel 7 im Rahmen eines umfangreichen Experiments zur Bewertung der erreichbaren Qualität bei der Trace-Link-Erfassung. Die Auswahl eines geeigneten Schwellwerts für die Trace-Link-Erfassung ist sehr wichtig, weil dieser unmittelbar die Qualität der erstellten Trace Links beeinflusst. Wenn der Schwellwert zu hoch gewählt wird, werden unter Umständen nicht alle relevanten Trace Links erfasst und wichtige Zusammenhänge bleiben unerkannt. Wenn der Schwellwert jedoch zu niedrig gewählt wird, können falsche Trace Links zwischen Elementen entstehen, die eigentlich nicht in Beziehung zueinander sind.

3.9. Erstellung von Skizzen

In Anlehnung an die Verwendung traditioneller analoger Whiteboards muss die Erstellung von Skizzen im AugIR barrierefrei und intuitiv sein. Insbesondere müssen auch nicht-technische Stakeholder Inhalte schnell und einfach bearbeiten und modifizieren können. Auf den elektronischen Whiteboards im AugIR stehen deshalb theoretisch unendlich große Zeichenflächen zur Verfügung, die sich frei in alle Richtungen verschieben und stufenlos zoomen lassen. Zur Interaktion mit diesen können drei verschiedene Werkzeuge verwendet werden:

Stift Mit einem speziellen Stift können Stakeholder auf die Fläche des elektronischen Whiteboards schreiben und Freihandskizzen erstellen. Die Haptik orientiert sich dabei an der Verwendung traditioneller Whiteboard-Marker. Stakeholder können Linienzüge zeichnen, indem sie den Stift auf der digitalen Zeichenfläche absetzen, umher bewegen und abschließend wieder hochheben. Auf die

gezeichneten Linienzüge werden die zuvor präsentierten Algorithmen zur automatischen Klassifizierung, Gruppierung, Tokenisierung und Erfassung von Trace Links angewendet.

Radierer Skizzierte Inhalte können mit einem speziellen Eingabegerät ausradiert werden. Dieses ist einem traditionellen Tafelschwamm nachempfunden, wodurch seine Verwendung auch für nicht-technische Stakeholder intuitiv und leicht verständlich ist. Während die meisten Tools zur Erstellung digitaler Freihandskizzen lediglich die komplette Entfernung gezeichneter Linienzüge erlauben, kommt im AugIR ein Algorithmus zum Einsatz, der ein pixelgenaues Radieren von Linienzügen ermöglicht. Zum einen können auf diese Weise detailliertere Skizzen erstellt werden. Zum anderen simuliert dies angemessener die Funktionsweise eines traditionellen analogen Whiteboard-Schwamms.

Finger Durch Verwendung ihrer Finger können Stakeholder die skizzierten Inhalte auf den Multi-Touch-Displays manipulieren. Sie können auf diese Weise zum Beispiel gezeichnete Elemente selektieren, kopieren und frei auf der Zeichenfläche verschieben sowie die Darstellung durch bestimmte Gesten beeinflussen. Hierdurch eröffnen sich neue Interaktionsmöglichkeiten, die auf traditionellen analogen Whiteboards nicht zur Verfügung stehen.

Mit den zuvor beschriebenen Eingabewerkzeugen können beliebige Freihandskizzen auf den elektronischen Whiteboards im AugIR erstellt und manipuliert werden.

Während viele verwandte Ansätze sich hierbei auf die Verwendung eines einzelnen großen Displays beschränken, kommen im AugIR mehrere miteinander verbundene elektronische Whiteboards zum Einsatz. Dies ermöglicht die gleichzeitige Betrachtung des Softwareprojekts aus unterschiedlichen Perspektiven, was insbesondere für Softwareingenieure und Designer nützlich ist [89]. Die selbe Skizze kann somit auf mehreren Whiteboards simultan geöffnet und von den Stakeholdern bearbeitet werden. Die skizzierten Inhalte werden hierbei in Echtzeit synchronisiert.

Um den Entwurf alternativer Lösungen zu erleichtern, kommt ein komplexes Undo-/Redo-System zum Einsatz, dass auf dem *Kommando-Entwurfsmuster* (engl. *Command Pattern*) der Gang of Four (GoF) [78] basiert. Alle ausführbaren Befehle wer-

den hierbei durch Kommando-Objekte gekapselt, wodurch sie beispielsweise effizient in eine Warteschlange eingereiht und rückgängig gemacht werden können.

Darüber hinaus lassen sich alle Inhalte auf Knopfdruck vervielfältigen, wodurch beliebig viele Kopien einer Skizze angefertigt und in unterschiedliche Richtungen weiterentwickelt werden können. Verschiedene Skizzen können dabei miteinander verglichen werden, indem sie auf dem selben elektronischen Whiteboard nebeneinander angeordnet oder simultan auf unterschiedlichen Whiteboards geöffnet werden.

Die skizzierten Inhalte werden dabei automatisch in regelmäßigen Abständen gespeichert, so dass sich die beteiligten Stakeholder vollständig auf den Modellierungsprozess konzentrieren können. Darüber hinaus lassen sich die erstellten Skizzen auf Knopfdruck als Vektorgrafiken exportieren, um sie in geeigneten Grafikprogrammen nachzubearbeiten oder direkt in entsprechenden Dokumentationen einzubinden.

Zur Annotierung der Inhalte werden in jedem Canvas a priori ausgewählte Annotationen als Symbole in einer Seitenleiste angezeigt. Diese können per Drag and Drop auf den gezeichneten Skizzenelementen platziert werden.

Das zuvor beschriebene Konzept ermöglicht bereits eine Erstellung von Process, Object und Integration Canvases, wie es auch in analogen Interaction Rooms möglich ist. Weil deren Syntax lediglich lose an UML-Diagramme angelehnt ist, werden sie auch auf den elektronischen Whiteboards im AugIR als Freihandskizzen erstellt, die bewusst informell und unvollständig bleiben können.

Hinsichtlich der in Abschnitt 1.3 geschilderten Probleme nehmen lediglich Feature und Interaction Canvas eine Sonderrolle ein. Diese werden deshalb in den folgenden Abschnitten im Detail gesondert betrachtet.

3.9.1. Feature Canvas

In Kapitel 2 wurde bereits deutlich, dass sich elektronische Whiteboards sinnvoll durch die Anbindung mobiler Endgeräte erweitern lassen. Diese können beispielsweise zur Betrachtung existierender Artefakte [221], zum Austausch von Informationen

[95], zur Erstellung neuer Inhalte [94, 177, 258] sowie als private Arbeitsflächen [94] genutzt werden.

Auch im AugIR erscheint der Einsatz von mobilen Endgeräten sinnvoll, um auf diese Weise Medienbrüche zu eliminieren.

Durch elektronische Whiteboards können bereits fast alle Interaction-Room-Canvases problemlos digitalisiert werden. Lediglich die Befüllung des Feature Canvas gestaltet sich bei ausschließlicher Verwendung von großen Multi-Touch-Displays schwierig: Die Interaction-Room-Methode sieht vor, dass Stakeholder zunächst in Einzelarbeit eine Reihe von Karteikarten mit Features beschriften, bevor diese an eine Metaplanwand geheftet und gemeinschaftlich diskutiert werden. Elektronische Whiteboards können zwar analoge Metaplanwände für diesen Anwendungszweck adäquat abbilden, unterstützen jedoch nicht sinnvoll die individuelle Erstellung von Feature-Karten durch zahlreiche Stakeholder zur gleichen Zeit.

Im AugIR kommen deshalb mobile Endgeräte zum Einsatz, die mit den elektronischen Whiteboards verbunden werden. Jeder Stakeholder erhält ein eigenes Tablet, auf dem er digitale Feature-Karten schreiben und verwalten kann. Nach der Erstellung der Feature-Karten können diese vom Tablet auf das Whiteboard übertragen und dort im digitalen Feature Canvas gemeinschaftlich sortiert, diskutiert und annotiert werden. Die entsprechenden Komponenten und die Anforderungen an diese werden im Folgenden beschrieben.

Erstellung und Übertragung digitaler Feature-Karten

Um die Erstellung digitaler Feature-Karten auch für nicht-technische Stakeholder möglichst intuitiv zu gestalten, muss sich dieser Prozess an der Erstellung analoger Karteikarten orientieren. Es kommen deshalb mobile Endgeräte mit Stifteingabe zum Einsatz, auf denen Stakeholder handschriftlich digitale Karten schreiben können. Hierzu steht für jede Karte eine begrenzte Zeichenfläche zur Verfügung, welche das Zeichnen von Linienzügen in verschiedenen Farben ermöglicht. Die grafische Benutzerschnittstelle der Zeichenfläche ist dabei bewusst simpel gestaltet, um an analoge Karteikarten zu erinnern.

Weil die Erstellung der Feature-Karten einem Brainstorming entspricht, entstehen die Karten in der Regel nicht in der gleichen Reihenfolge in der sie später präsentiert werden. Stakeholder müssen deshalb zudem in der Lage sein, die erstellten Karten auf den mobilen Geräten zu sortieren und in einer frei wählbaren Reihenfolge nacheinander an das elektronische Whiteboard zu übertragen.

Die mobile Anwendung zur Erstellung digitaler Feature-Karten bietet deshalb zwei verschiedene Ansichten, die in Abbildung 3.9 skizziert sind: Eine Übersicht erlaubt die Erstellung neuer Karten und zeigt alle bisher auf dem mobilen Gerät erstellten Exemplare in einem zweidimensionalen Raster an. Einzelne Karten können von hier aus in beliebiger Reihenfolge an das elektronische Whiteboard übertragen oder vom Gerät gelöscht werden. Durch Antippen einer Karte in der Übersicht öffnet sich die Detailansicht, in der die ausgewählte Karte bearbeitet werden kann. Um Stakeholder zu ermuntern, sich bei der Formulierung auf das Wesentliche zu konzentrieren, ist die Größe dieser Zeichenfläche – anders als auf den elektronischen Whiteboards – auf die Displaygröße des mobilen Endgerätes beschränkt.

Darstellung und Verwaltung digitaler Feature-Karten

Nachdem die digitalen Karten auf den mobilen Endgeräten erstellt und von dort aus an die elektronischen Whiteboards übertragen wurden, werden diese im Feature Canvas dargestellt. Wie in Abbildung 3.10 illustriert wird hierbei jede Karte durch ein Rechteck repräsentiert, welches den handschriftlich geschriebenen Text enthält.

Der Feature Canvas stellt eine theoretisch unendlich große Fläche zur Verfügung, die beliebig in alle Richtungen verschoben und stufenlos gezoomt werden kann. Die erstellten Karten können auf dieser Fläche umherbewegt und frei angeordnet werden, um diese beispielsweise nach Themen oder inhaltlichen Schwerpunkten zu sortieren.

Darüber hinaus kann für jede Karte das eindeutige Kürzel des Stakeholders eingeblendet werden, der diese Karte auf seinem Tablet geschrieben hat. Somit muss diese Information vom Schriftführer nicht mehr explizit dokumentiert werden und

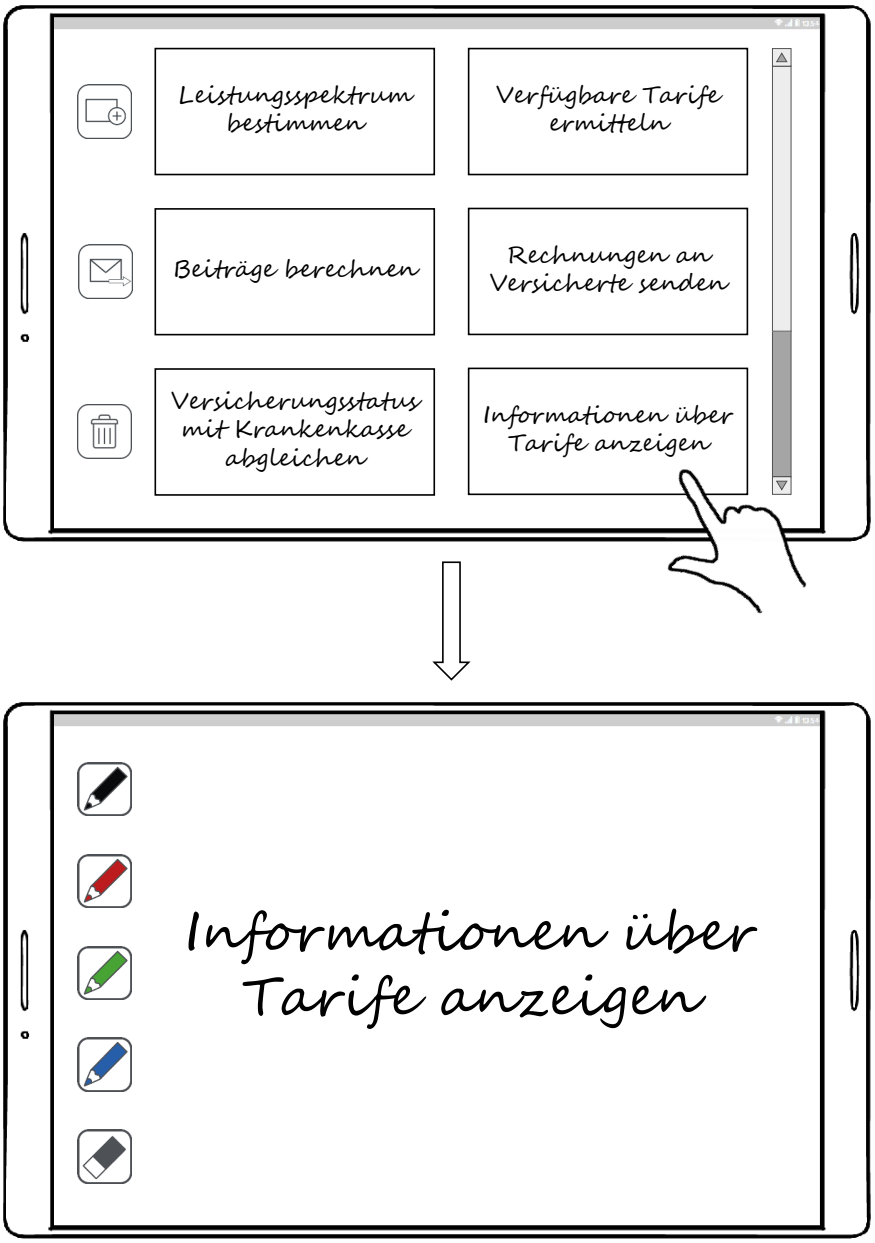

Abbildung 3.9.: Alle auf einem mobilen Endgerät erstellten Karten werden in einem zweidimensionalen Raster angezeigt und können in beliebiger Reihenfolge an das elektronische Whiteboard übertragen werden. Durch Antippen einer Karte wird diese zur Bearbeitung geöffnet.

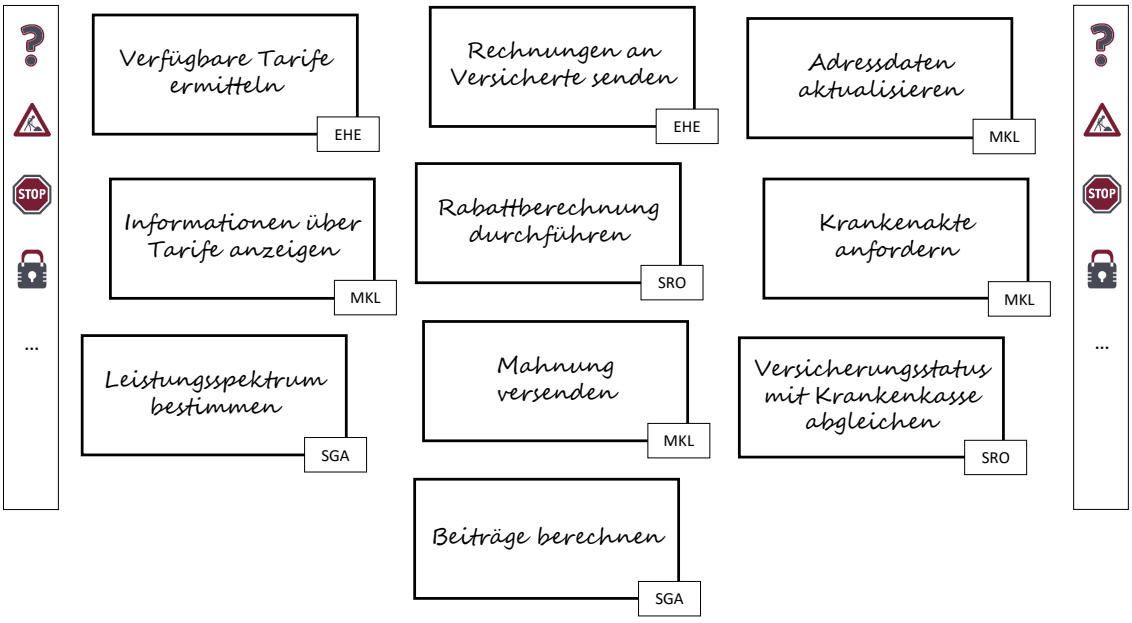

Abbildung 3.10.: Nach der Übertragung der digitalen Karten können diese auf dem Feature Canvas frei angeordnet und sortiert sowie per Drag and Drop mit Annotationen versehen werden. Optional lässt sich für jede Karte das Kürzel des Autors einblenden.

geht darüber hinaus auch im Nachhinein sofort aus der Betrachtung des Canvas hervor. Dies erscheint nützlich, um zum Beispiel nach einem Workshop leicht einen passenden Ansprechpartner für ein bestimmtes Feature zu identifizieren.

Genau wie alle anderen Canvases kann auch der Feature Canvas von den Stakeholdern annotiert werden. Hierzu steht eine Seitenleiste zur Verfügung, die eine a priori ausgewählte Menge an Annotationssymbolen bereitstellt. Diese können angetippt und per Drag and Drop auf den gewünschten Karten platziert werden. Um möglichst vielen Stakeholdern gleichzeitig die Möglichkeit zur Annotierung der Inhalte zu geben, wird die Seitenleiste sowohl am linken als auch am rechten Rand des Whiteboards eingeblendet. Darüber hinaus kann der Feature Canvas auch auf mehreren Whiteboards im AugIR gleichzeitig geöffnet und gemeinschaftlich annotiert werden.

3.9.2. Interaction Canvas

Statische UI-Skizzen alleine reichen oft nicht aus, um Stakeholdern ein adäquates Gefühl für das Look and Feel einer Anwendung zu vermitteln [178], weil die Beschreibung des dynamischen Verhaltens einer grafischen Benutzerschnittstelle häufig wesentlich schwieriger ist als die ihres optischen Erscheinungsbildes [174]. Aus diesem Grund erlauben die meisten der in Abschnitt 2.2.2 vorgestellten Werkzeuge zur informellen Skizzierung von grafischen Benutzeroberflächen die interaktive Ausführung der skizzierten Inhalte.

Durch die stetig wachsende Verbreitung von berührungssensitiven Geräten wie Smartphones, Tablets, elektronischen Whiteboards und Tabletops wird darüber hinaus bei der Verhaltensbeschreibung von Anwendungen auch die Betrachtung von Gesten immer wichtiger [104]. Hinsichtlich der User Experience muss beachtet werden, ob Benutzer die definierten Gesten als natürlich, verständlich und einfach anwendbar empfinden [252].

Während viele verwandte Ansätze zwar einen vordefinierten Satz einfacher Gesten bereitstellen, die zur Verhaltensbeschreibung einer skizzierten Anwendung genutzt

werden können, ist die Verwendung komplexer nicht-standardisierter Gesten jedoch häufig nicht ohne Weiteres möglich [118, 252].

Um diesen Beobachtungen Rechnung zu tragen, stellt der AugIR einen entsprechenden Simulationsmodus für den Interaction Canvas zur Verfügung, der die dynamische Ausführung von Transitionen zwischen Skizzen ermöglicht. Darüber hinaus bietet er die Möglichkeit, benutzerdefinierte Gesten zur Beschreibung des dynamischen Verhaltens zu spezifizieren. Hierdurch sollen Stakeholder ein besseres Verständnis der Dialogflüsse erhalten und Diskussionen zur Interaktivität der Benutzeroberfläche unterstützt werden.

Im Folgenden wird zunächst diskutiert, auf welche Weise sich Gesten im AugIR spezifizieren lassen. Danach wird erläutert, wie digitale UI-Skizzen auf einem Storyboard miteinander verknüpft und anschließend interaktiv in einem Simulationsmodus ausgeführt werden können.

Spezifikation von Gesten

Existierende Ansätze zur Beschreibung von Gesten lassen sich grundsätzlich in vier Kategorien einteilen [104]:

- *Template-basierte Ansätze* [9, 240, 253] erlauben die Aufzeichnung von Beispielgesten, um diese zur Interpretation von Interaktionen heranzuziehen. Sie sind hierbei jedoch auf die Angabe der geometrischen Form einer Geste beschränkt. Template-basierte Ansätze sind in der Regel leicht zu implementieren und insbesondere für nicht-technische Stakeholder einfach zu verstehen und anzuwenden. Weil sie jedoch neben der Form einer Geste keine Angabe zusätzlicher Parameter ermöglichen und selten Multi-Touch-Eingaben unterstützen, eignen sich Template-basierte Erkenner nur für simple Gesten. Eine Definition komplexer Gesten ist mit ihnen kaum sinnvoll möglich.
- *Textuelle Notationen* [108] beschreiben Gesten durch eine Grammatik und einen Satz von Schlüsselwörtern, die mit Hilfe von Operatoren in formale Gestenspezifikationen überführt werden können. Dies kann beispielsweise durch

die Verknüpfung atomarer Einzelgesten [117] oder durch die Verwendung regulärer Ausdrücke [120] geschehen. Textuelle Notationen sind grundsätzlich mächtiger als Template-basierte Ansätze und ermöglichen die Spezifikation einer Vielzahl von komplexen Gesten, die mit Template-basierten Ansätzen nicht realisierbar wären. Jedoch ist die Definition von Gesten durch textuelle Notation aufwändiger und das Resultat ist insbesondere mit steigender Komplexität der Gesten schwerer lesbar. Zudem müssen Stakeholder zunächst die Notation und die verfügbaren Schlüsselwörter erlernen, bevor sie diese zielgerichtet einsetzen können.

- *Grafische Notationen* [79, 119] verwenden eine Menge von visuellen Symbolen, aus denen Gestenspezifikationen zusammengesetzt werden können. Ihr Ziel ist die Reduktion der Komplexität bei der Definition von Gesten, indem sie textuelle Beschreibungen durch visuelle Elemente ersetzen. Hierzu werden häufig geometrische Formen wie Rechtecke, Kreise und Pfeile verwendet, die jeweils bestimmte Aspekte einer Geste repräsentieren. Obwohl grafische Notationen auf den ersten Blick zunächst einfacher verständlich zu sein scheinen als textuelle Notationen, haben sie den gleichen Nachteil: Sie stellen einen großen Satz unterschiedlicher Symbole und Operatoren zur Verfügung, die von den Stakeholdern zunächst erlernt werden müssen. Insbesondere die Definition komplexer Gesten ist daher mit grafischen Notationen ähnlich schwierig wie mit textuellen Notationen.

- *Hybride Notationen* [151] kombinieren die vorgenannten Ansätze, um hierdurch die Vorteile verschiedener Herangehensweisen zu nutzen. Gesten können durch Aufzeichnung von Beispielen definiert und durch Angabe zusätzlicher Eigenschaften konkreter spezifiziert werden. Hybride Notationen können somit die Spezifikation und Diskussion von Gesten wesentlich vereinfachen. Trotz ihrer offensichtlichen Vorteile werden hybride Notationen nur äußerst selten verwendet [104].

Die vorausgegangene Betrachtung unterschiedlicher Ansätze zur Beschreibung von Gesten legt nahe, dass die Verwendung einer hybriden Notation zahlreiche Vorteile

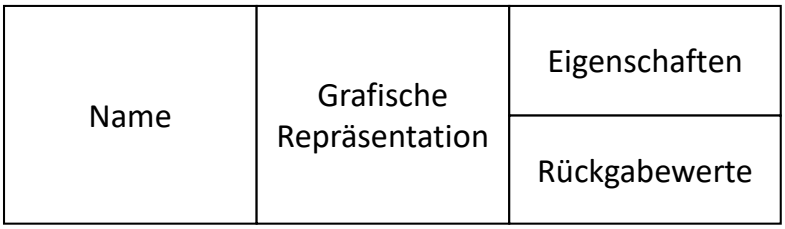

Abbildung 3.11.: Aufbau von GestureCards (in Anlehnung an [104]).

mit sich bringt. Aus diesem Grund wird im AugIR ebenfalls ein hybrider Ansatz verfolgt, der für Stakeholder leicht verständlich ist und dennoch die präzise Spezifikation von Gesten ermöglicht. Eine Geste lässt sich somit im AugIR durch das Zeichnen einer Freihandform definieren. Zusätzlich kann diese Geste durch Angabe bestimmter Eigenschaften, wie beispielsweise der erwarteten Ausführungsgeschwindigkeit, präzisiert werden. Die Eigenschaften können dabei in einem grafischen Editor ausgewählt und miteinander kombiniert werden, um auch nicht-technischen Stakeholdern die einfache und intuitive Spezifikation von Gesten zu ermöglichen.

Zur Notation werden hierzu die von Hesenius et al. [103, 104] vorgestellten *GestureCards* verwendet. Hierbei handelt es sich um eine hybride Repräsentation von Gesten, die grafische und textuelle Elemente miteinander verbindet.

GestureCards werden in Form von analogen oder digitalen Karteikarten notiert, deren Aufbau in Abbildung 3.11 skizziert ist:

Ganz links steht der Name der definierten Geste. Dieser muss eindeutig sein, damit sie von anderen GestureCards referenziert werden kann, um beispielsweise komplexere Gesten aus mehreren Einzelgesten zusammenzusetzen.

Daneben ist eine grafische Repräsentation der gezeichneten Beispielgeste abgebildet, welche die Form der Geste beschreibt. Analoge GestureCards enthalten an dieser Stelle ein statisches Bild der Geste, wobei jeder Zug durch eine Linie dargestellt ist, deren Richtung durch eine Pfeilspitze angezeigt wird. Im AugIR wird das statische Bild durch eine Animation ersetzt, welche die Bewegung der Finger auf der Touch-Oberfläche simuliert. Hierdurch kann die Ausführung der Geste im Vergleich zur

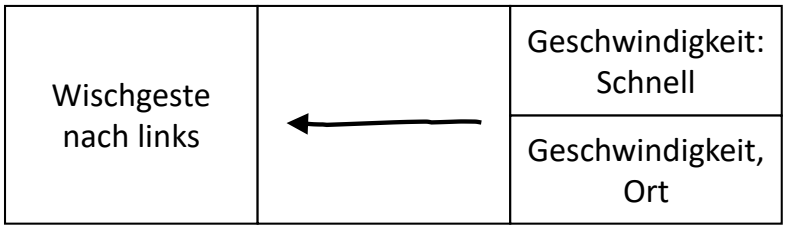

Abbildung 3.12.: Beispiel für eine GestureCard, die eine schnelle Wischgeste nach links definiert und im Erkennungsfall Informationen zu ihrer tatsächlichen Ausführungsgeschwindigkeit und Position zurückgibt (in Anlehnung an [104]).

analogen Variante noch intuitiver und leichter verständlich visualisiert werden.

Rechts oben werden die Eigenschaften aufgelistet, die zusätzlich zur geometrischen Form für diese Geste angegeben wurden. GestureCards definieren hierzu insgesamt 19 mögliche Eigenschaften, wie beispielsweise die maximale Dauer einer Geste, ihre durchschnittliche Ausführungsgeschwindigkeit und das verwendete Eingabewerkzeug.

Rechts unten werden optional die Eigenschaften aufgelistet, die vom Gestenerkenner an die Applikation zurückgegeben werden können, wenn die entsprechende Geste erkannt wurde. Auf diese Weise kann die Anwendung bestimmte Informationen auswerten, weiterverarbeiten und geeignet darauf reagieren.

Abbildung 3.12 zeigt ein einfaches Beispiel für eine GestureCard, die eine nach links gerichtete Wischgeste spezifiziert. Die Geschwindigkeitseigenschaft zeigt dabei an, dass die Geste in einem hohen Tempo auszuführen ist. Langsame Wischgesten werden folglich ignoriert. Die GestureCard liefert nach ihrer Erkennung die tatsächliche Ausführungsgeschwindigkeit der Geste sowie deren Position auf der Touch-Oberfläche zurück.

GestureCards können darüber hinaus auch miteinander kombiniert werden, um auf diese Weise komplexere Gesten zu definieren, die aus mehreren einfachen Einzelgesten bestehen. Dies wird im Detail in der Arbeit von Hesenius et al. [104] beschrieben.

Verwaltung und Verknüpfung von UI-Skizzen

Nach ihrer Erstellung werden die UI-Skizzen auf dem Interaction Canvas dargestellt. Dieser entspricht im Wesentlichen einem digitalen Storyboard, das die UI-Skizzen in Form rechteckiger Vorschaubilder anzeigt. Die Skizzen können auf dem Interaction Canvas beliebig verschoben und angeordnet werden. Wie alle anderen Canvases verfügt auch der Interaction Canvas hierzu über eine theoretisch unendlich große Fläche, die frei in alle Richtungen verschoben und stufenlos gezoomt werden kann. Durch Antippen einer UI-Skizze wird diese auf dem aktuellen Whiteboard zur Bearbeitung im Vollbildmodus geöffnet.

Darüber hinaus erlaubt der Interaction Canvas insbesondere die Verknüpfung von UI-Skizzen, um auf diese Weise Transitionen zwischen Dialogen zu modellieren. Die Struktur des Interaction Canvas entspricht hierbei einem Graph und ist in Abbildung 3.13 dargestellt:

Jedes Projekt besitzt einen Interaction Canvas, der eine beliebige Anzahl von UI-Skizzen enthalten kann. UI-Skizzen können auf dem Interaction Canvas durch *Gesture Links* miteinander verknüpft werden, um einen Übergang zwischen diesen zu modellieren. Ein Gesture Link verbindet hierbei stets genau zwei UI-Skizzen miteinander, wobei jede UI-Skizze mit beliebig vielen anderen verknüpft sein kann. Eine UI-Skizze ist dabei die Quelle, in der die Geste ausgeführt werden muss. Die andere UI-Skizze definiert das Ziel, zu dem bei erfolgreicher Erkennung der Geste navigiert werden soll.

Jeder Gesture Link referenziert eine GestureCard, welche die Geste beschreibt, die zur Auslösung der entsprechenden Transition ausgeführt werden muss. GestureCards können beliebig oft wiederverwendet und von mehreren Gesture Links referenziert werden. Wie im vorherigen Abschnitt erläutert, bestehen sie aus einer Reihe von Eigenschaften und werden durch die grafische Repräsentation einer Beispielgeste beschrieben. Darüber hinaus können GestureCards ineinander verschachtelt werden, um auf diese Weise komplexe Gesten aus mehreren einfachen Gesten zusammenzusetzen.

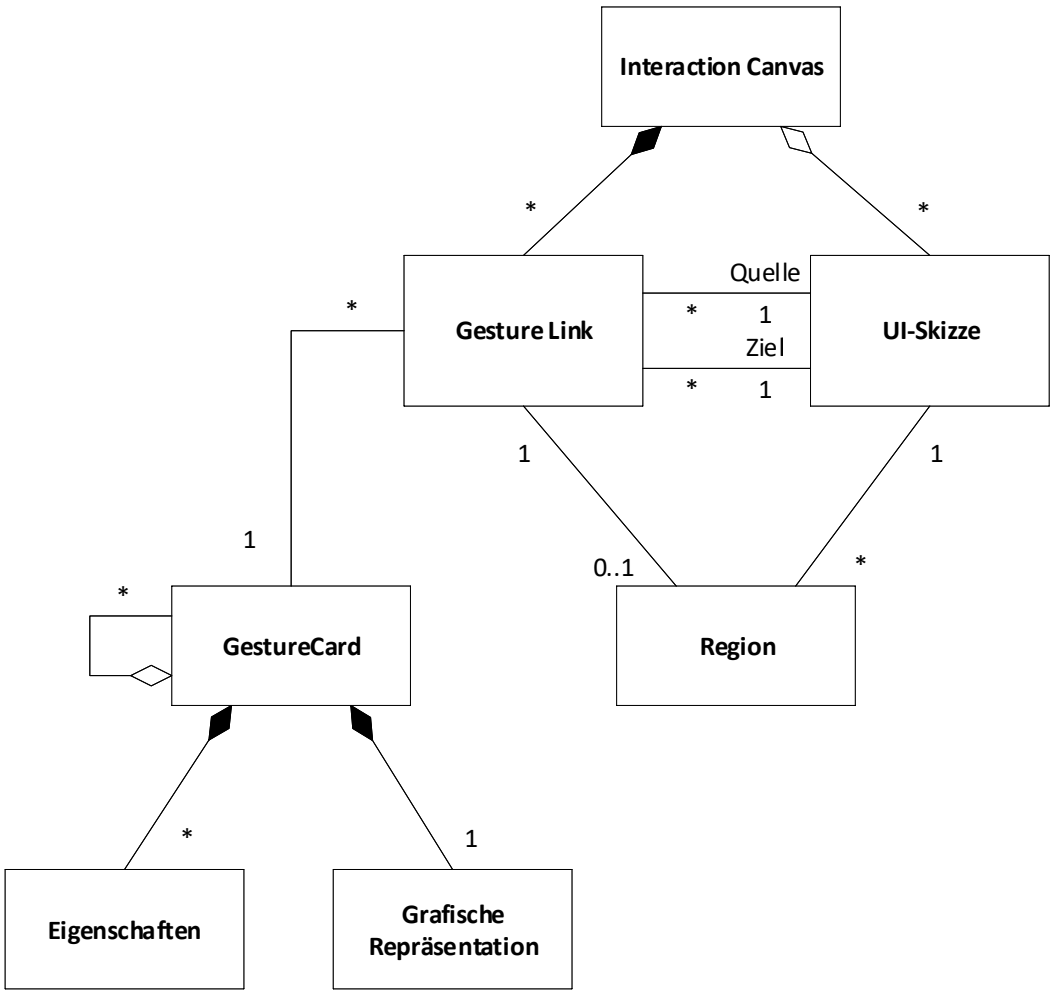

Abbildung 3.13.: Struktur des Interaction Canvas zur Verwaltung und Verknüpfung von UI-Skizzen im AugIR.

Für jeden Gesture Link kann außerdem optional eine Region in einer UI-Skizze spezifiziert werden. Diese gibt an, in welchem Bereich der Skizze die zugehörige GestureCard gültig sein soll. Hiermit lassen sich Gesten entweder global definieren oder auf ein bestimmtes Areal einer Skizze einschränken.

Um einen Gesture Link zwischen zwei UI-Skizzen zu definieren, können Stakeholder diese mit einem Whiteboard-Stift auf dem Storyboard durch eine gerade Linie verbinden. Daraufhin öffnet sich ein Dialog zur Spezifikation der gewünschten Geste, die diese Transition auslösen soll.

Nach der Definition eines solchen Dialogübergangs zeigt der Interaction Canvas einen Verbindungspfeil zwischen den beiden verknüpften UI-Skizzen an und annotiert diesen mit der entsprechenden GestureCard. Auf diese Weise sind die spezifizierten Benutzerinteraktionen auch für nicht-technische Stakeholder intuitiv verständlich und gehen unmittelbar aus der Betrachtung des Storyboards hervor.

Abbildung 3.14 illustriert dies an einem einfachen Beispiel: Zwei UI-Skizzen, die eine Auflistung von Elementen auf einem mobilen Gerät anzeigen, wurden über einen Gesture Link miteinander verknüpft. Durch Ausführung einer Wischgeste nach oben wird der Inhalt der skizzierten Liste nach oben verschoben. Dies wird durch die linke GestureCard angezeigt. Die rechte GestureCard definiert eine Wischgeste in die umgekehrte Richtung, welche den Listeninhalt wieder nach unten schiebt und zum Anfang der Liste zurück scrollt. Die verwendete Eigenschaft auf beiden GestureCards zeigt an, dass die Geste nur erkannt werden soll, wenn sie auf dem skizzierten Listenelement ausgeführt wird.

Interaktive Ausführung von UI-Skizzen

Die auf dem Interaction Canvas spezifizierten Dialogübergänge können in einem speziellen Simulationsmodus interaktiv ausgeführt werden. Dies gibt den Stakeholdern Gelegenheit, die definierten Gesten live zu testen. Hierzu muss eine UI-Skizze im Vollbildmodus geöffnet und der Simulationsmodus gestartet werden. Während die

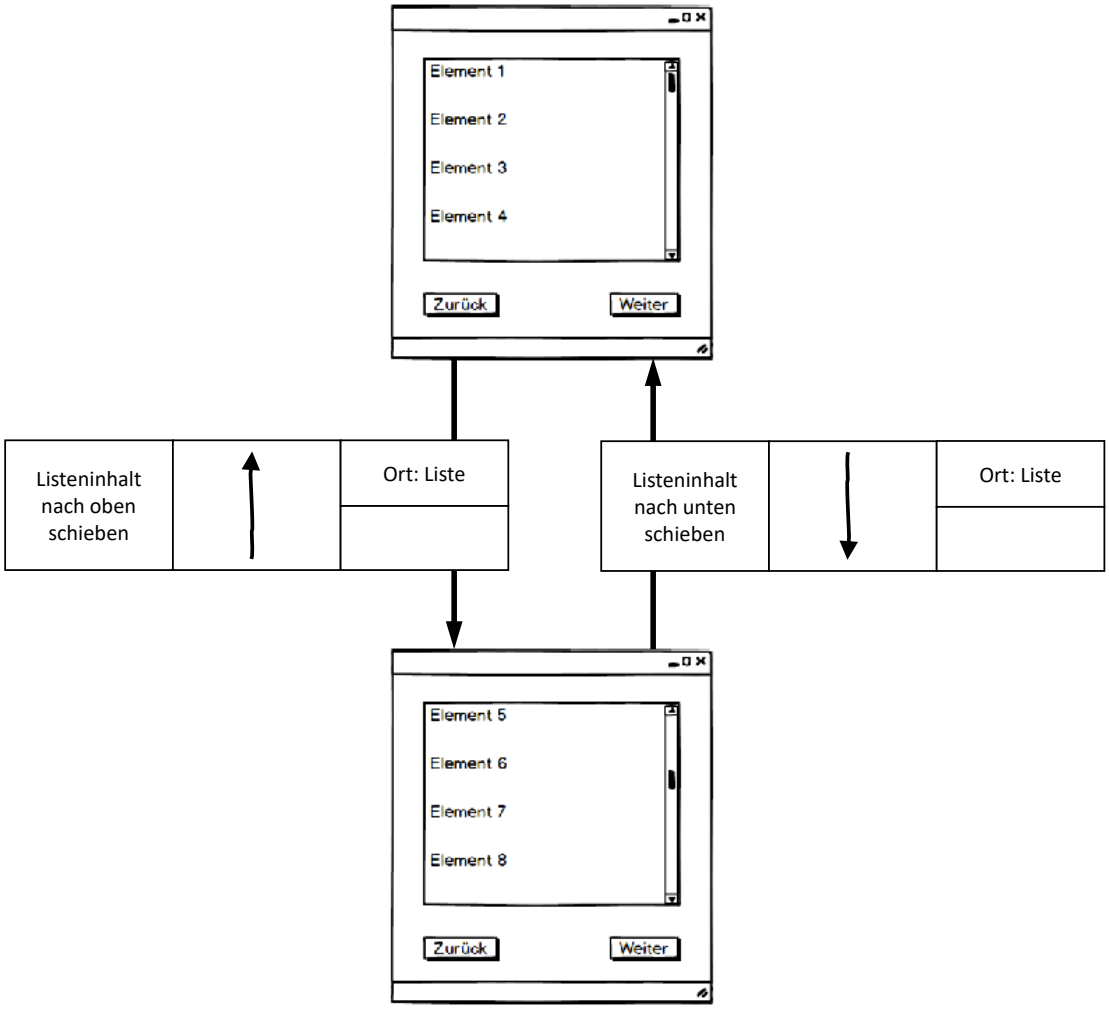

Abbildung 3.14.: Beispielhafter Ausschnitt aus einem Interaction Canvas: Zwei UI-Skizzen werden über Gesture Links miteinander verknüpft. Der Inhalt der skizzierten Liste kann über eine Wischgeste nach oben oder unten in die entsprechende Richtung verschoben werden.

Simulation läuft, können die Skizzen nicht bearbeitet werden und sämtliche Interaktionen mit der Oberfläche des elektronischen Whiteboards werden an den Gestenerkenner weitergeleitet. Sobald eine ausgeführte Geste in der aktuell dargestellten Skizze erkannt wurde, tauscht der AugIR diese unmittelbar durch die UI-Skizze aus, die im zugehörigen Gesture Link als Ziel angegeben wurde. Dies simuliert einen dynamischen Übergang von einer Skizze zu einer anderen. Der Simulationsmodus kann dabei jederzeit pausiert werden, um eventuell notwendige Anpassungen an den definierten Gesten vorzunehmen.

Damit auch Stakeholder ohne umfangreiches Projektwissen und ohne detaillierte Kenntnis der bereits definierten Gesten den Simulationsmodus intuitiv und ohne aufwändige Vorbereitung verwenden können, zeigt dieser stets alle verfügbaren Interaktionsmöglichkeiten für den aktuellen Kontext an. Dies ist in Abbildung 3.15 illustriert: Um die aktuell geöffnete UI-Skizze herum werden Vorschaubilder aller Skizzen angeordnet, die mit dieser über einen Gesture Link verbunden sind. Auf diese Weise ist stets ersichtlich, wohin von der aktuellen Position aus navigiert werden kann. Unter jedem Vorschaubild ist darüber hinaus die zugehörige GestureCard abgebildet, um alle Gesten aufzulisten, die im aktuellen Kontext verfügbar sind. Wenn die definierten Gesten zudem auf einen spezifischen Bereich der angezeigten Skizze eingeschränkt wurden, so ist dieser zusätzlich farblich markiert und über eine Linie visuell mit dem entsprechenden Vorschaubild verbunden.

3.10. Verwaltung von Skizzen und Verknüpfungen

In Abschnitt 2.2.3 wurden unterschiedliche Herangehensweisen zur Verwaltung digital erstellter Freihandskizzen präsentiert. Im AugIR wird hierzu ein Graph als Datenstruktur verwendet, der im Folgenden als *Navigationsgraph* bezeichnet wird. Abbildung 3.16 visualisiert dessen Struktur.

Wie jeder Graph besteht auch der Navigationsgraph aus Kanten und Knoten. Hierbei wird zwischen *Gruppenknoten* und *Skizzenknoten* unterschieden: Gruppenknoten dienen zur Strukturierung und Gruppierung der Inhalte. Sie können beispielsweise verschiedene Aspekte einer Komponente des zu entwickelnden Softwaresystems

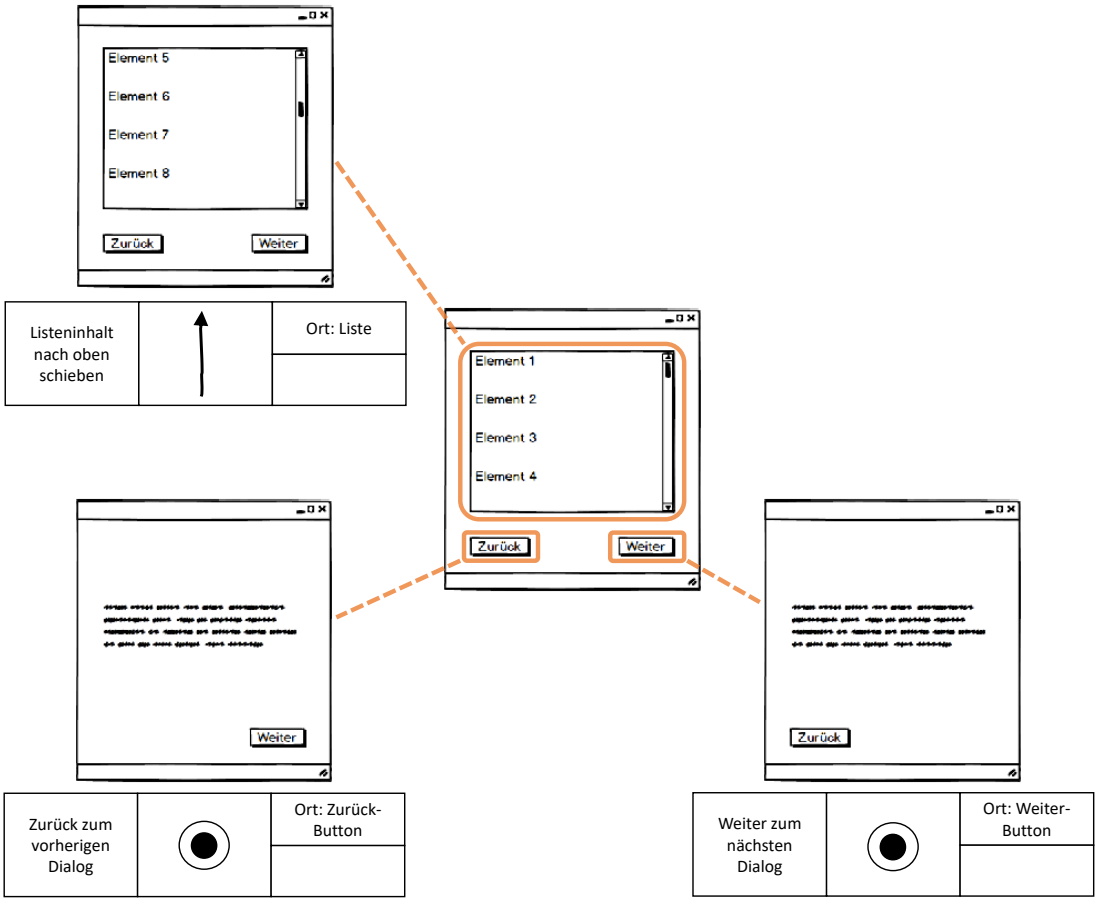

Abbildung 3.15.: Im Simulationsmodus werden um die aktuell geöffnete UI-Skizze herum Vorschaubilder aller verknüpften Skizzen angeordnet. Die GestureCard unter jedem Vorschaubild zeigt an, mit welcher Geste zu der entsprechenden Skizze navigiert werden kann. Die grafischen Repräsentationen auf den beiden unteren GestureCards stehen für ein einfaches kurzes Antippen.

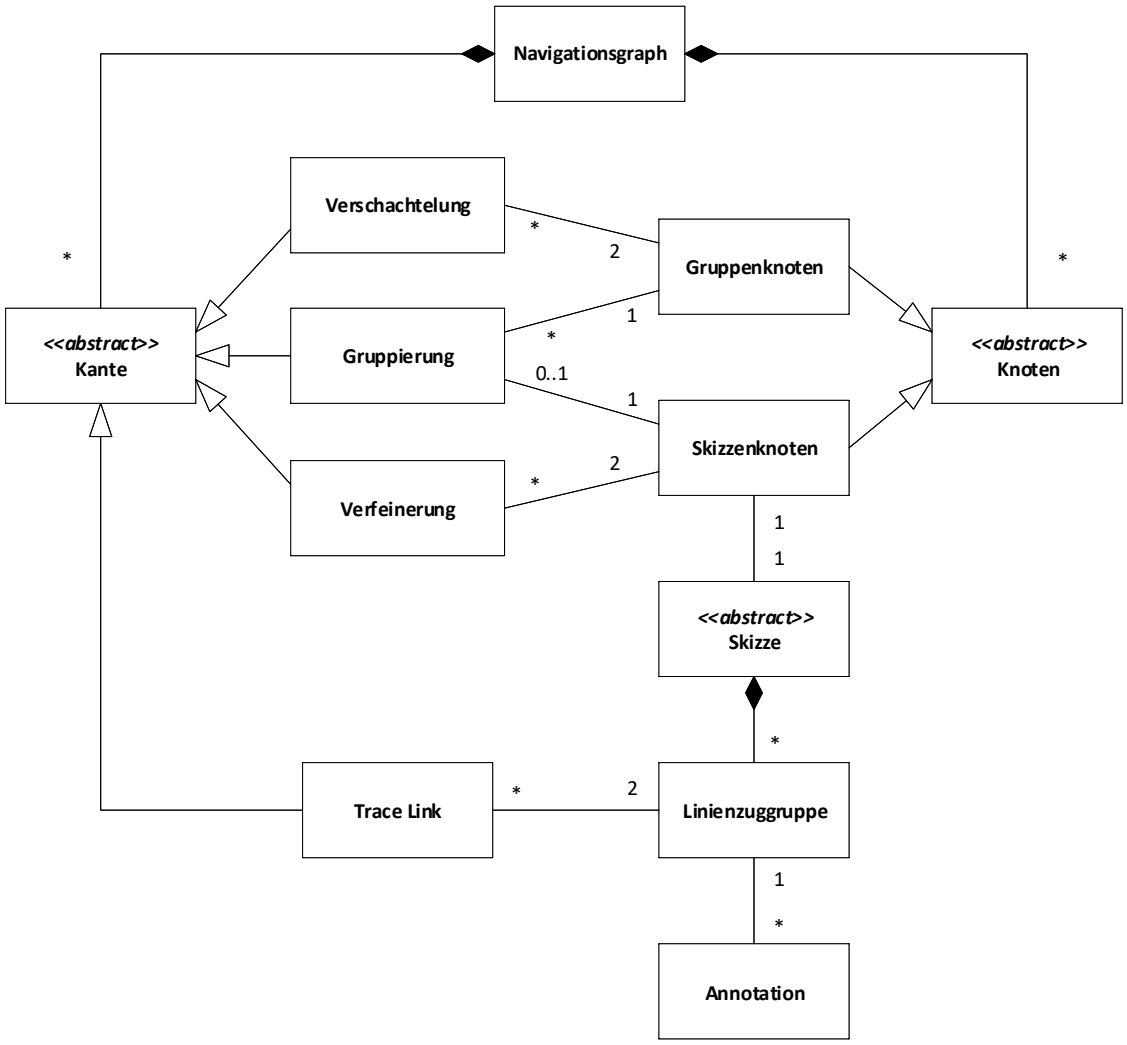

Abbildung 3.16.: Struktur des Navigationsgraphen im AugIR.

oder eine Reihe von inhaltlich zusammenhängenden Interaction-Room-Canvases zusammenfassen. Demgegenüber repräsentieren Skizzenknoten die im AugIR erstellten Freihandskizzen. Dabei stellt jeder Skizzenknoten genau eine Skizze dar, die jeweils einem Interaction-Room-Canvas entspricht.

Jeder Gruppenknoten kann eine beliebige Anzahl von Skizzenknoten gruppieren, wobei jeder Skizzenknoten maximal einer Gruppierung zugeordnet werden kann. Um eine strukturierte Modellierung des Systems zu ermöglichen, können Gruppenknoten darüber hinaus beliebig tief ineinander verschachtelt werden und weitere Gruppenknoten enthalten.

In Abschnitt 1.3.1 wurde bereits diskutiert, dass insbesondere umfangreiche Diagramme, wie beispielsweise komplexe Geschäftsprozesse, in der Praxis zur Reduktion der Komplexität häufig in Unterdiagrammen verfeinert werden. Der Navigationsgraph unterstützt deshalb eine Abbildung dieser Beziehungen, indem er die Verfeinerung einer Skizze durch eine andere ermöglicht. Wie diese Verfeinerungsbeziehungen Stakeholder bei der vertikalen Navigation zwischen Skizzen unterstützen können, wird in Abschnitt 3.11.1 diskutiert.

Jede erstellte Freihandskizze kann beliebig viele Linienzüge enthalten, die wie in Abschnitt 3.5 beschrieben automatisch zu logisch zusammengehörigen Linienzuggruppen zusammengefasst werden.

Mit Hilfe des in Abschnitt 3.8 vorgestellten Traceability-Algorithmus können Trace Links zwischen diesen Gruppen erfasst und nutzbar gemacht werden. Jeder Trace Link verknüpft hierbei stets genau zwei Linienzuggruppen unterschiedlicher Skizzen miteinander. Wie diese Trace Links visualisiert und verwendet werden können, wird in Abschnitt 3.11.2 erläutert.

Darüber hinaus können die Linienzuggruppen mit den in Abschnitt 1.2 beschriebenen Annotationen versehen werden.

Abbildung 3.17 illustriert den Aufbau eines Navigationsgraphen anhand eines einfachen Beispiels. Die abgerundeten Rechtecke repräsentieren Gruppenknoten, während

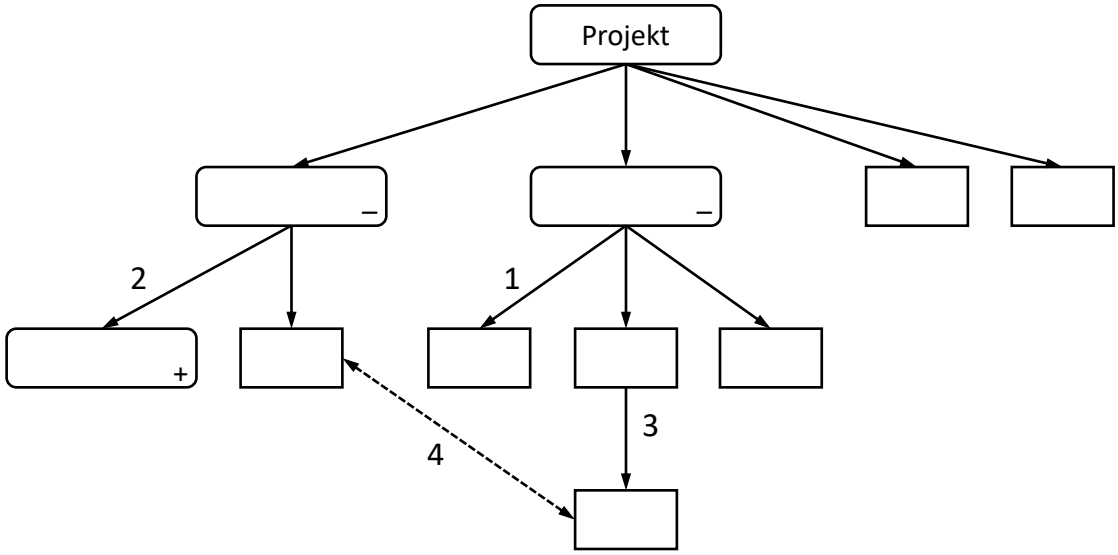

Abbildung 3.17.: Beispiel für den Aufbau eines Navigationsgraphen.

die gewöhnlichen Rechtecke Skizzenknoten darstellen. Die durchgezogenen Verbindungspfeile stellen Gruppierungen, Verschachtelungen und Verfeinerungen dar, während der gestrichelte Pfeil einen Trace Link zwischen zwei Linienzuggruppen in den entsprechenden Skizzen anzeigt. Die Pfeilspitzen deuten hierbei jeweils die Richtung der Beziehung an: Bei Gruppierungen, Verschachtelungen und Verfeinerungen zeigen sie stets auf das Kind-Element, während Trace Links immer bidirektional sind. Bei dem obersten Knoten handelt es sich um einen Metaknoten, der das eigentliche Projekt mitsamt seinen Inhalten repräsentiert.

Die Zahlen an den Verbindungspfeilen in Abbildung 3.17 kennzeichnen die unterschiedlichen Arten von Kanten, die im Navigationsgraph existieren können:

1 – Gruppierung Skizzenknoten können unterhalb von Gruppen platziert werden, um eine strukturierte Modellierung des Softwareprojekts zu ermöglichen. Auf diese Weise können Stakeholder zum Beispiel inhaltlich oder logisch zusammenhängende Skizzen zusammenfassen.

2 – Verschachtelung Gruppenknoten können beliebig tief ineinander verschachtelt werden, um das Projekt hierarchisch zu strukturieren. Die Plus- und Minus-

zeichen auf diesen Knoten deuten dabei an, dass sie dynamisch aus- und eingeklappt werden können, um zielgerichtet darunterliegende Details anzuzeigen oder zu verbergen.

3 – Verfeinerung Skizzen können durch andere Skizzen verfeinert werden, um somit die Visualisierung und Modellierung komplexer Sachverhalte über mehrere Abstraktionsebenen hinweg zu ermöglichen und die vertikale Navigation zwischen den Inhalten zu unterstützen.

4 – Trace Link Linienzuggruppen verschiedener Skizzen können über Trace Links miteinander verbunden werden, um inhaltliche Zusammenhänge und Abhängigkeiten aufzudecken sowie die horizontale Navigation zwischen den verknüpften Elementen zu unterstützen.

Die Verwendung einer Graph-Datenstruktur hat gegenüber verwandten Ansätzen wie beispielsweise eines dynamischen Filmstreifens [219, 220] oder eines festen zweidimensionalen Gitters [156] mehrere Vorteile:

Zum einen ermöglicht sie eine hierarchische Darstellung der Inhalte und der Beziehungen zwischen diesen. Dies ist insbesondere zur Unterstützung der vertikalen Navigation nützlich und erscheint überdies als natürliche Art, Verfeinerungen zwischen Skizzen über mehrere Abstraktionsebenen hinweg abzubilden.

Die virtuelle Fläche des Navigationsgraphen ist theoretisch unendlich groß und kann frei in alle Richtungen verschoben und stufenlos gezoomt werden. Jeder Skizzenknoten zeigt hierbei eine Vorschau der zugehörigen Freihandskizze, deren Inhalt auf diese Weise vergrößert oder verkleinert dargestellt werden kann. Hierdurch können Stakeholder die Ansicht flexibel dem aktuellen Problemkontext anpassen.

Die Verwendung mehrerer Whiteboards ermöglicht darüber hinaus die gleichzeitige Visualisierung des Problemkontextes durch Anzeige des Navigationsgraphen auf einem Whiteboard und der Darstellung problemspezifischer Details durch Anzeige spezifischer Skizzen auf den verbundenen Whiteboards. Diese simultane Visualisierung von mikroskopischen und makroskopischen Inhalten des betrachteten Softwareprojekts ist insbesondere für die Bearbeitung komplexer Aufgaben hilfreich [43].

Weiterhin erlaubt die Verwendung von Gruppenknoten eine Strukturierung des Projekts, weil logisch oder inhaltlich zusammengehörige Elemente zusammengefasst werden können. Durch dynamisches Auf- und Zuklappen von Gruppen können die entsprechenden Inhalte dabei jederzeit zielgerichtet ein- und ausgeblendet werden. Hierdurch können Stakeholder irrelevante Details verbergen und sich besser auf den aktuellen Problemkontext fokussieren.

Obwohl der Navigationsgraph die Modellierung einer hierarchischen Struktur ermöglicht und nahelegt, müssen Stakeholder die erstellten Freihandskizzen nicht in einer vorgegebenen Art und Weise strukturieren. Sowohl Gruppen- als auch Skizzenknoten können vollständig unabhängig voneinander erstellt und frei auf dem elektronischen Whiteboard angeordnet werden. Der Navigationsgraph muss nicht zusammenhängend sein.

Durch Antippen eines Skizzenknotens wird die entsprechende Skizze auf dem aktuellen Whiteboard zur Bearbeitung im Vollbildmodus geöffnet. Auf diese Weise können Stakeholder im AugIR jederzeit dynamisch den angezeigten Canvas-Typ auf jedem Display wechseln.

3.11. Navigation zwischen Skizzen

Im folgenden Abschnitt wird erläutert, wie Verfeinerungen und Trace Links visuell dargestellt und zur vertikalen und horizontalen Navigation zwischen skizzierten Inhalten verwendet werden können.

3.11.1. Vertikale Navigation

Im AugIR können beliebig viele Freihandskizzen erzeugt und frei im Navigationsgraph platziert werden. Die Skizzen müssen hierzu nicht zwingend miteinander verbunden sein sondern können unabhängig voneinander bleiben. Um jedoch eine hierarchische Struktur innerhalb des Projekts zu ermöglichen, können darüber hinaus

Abbildung 3.18.: Die Verfeinerung eines Elements – beispielsweise einer komplexen Aktion in einer Prozessskizze – wird durch einen stilisierten Dreizack angezeigt.

Verfeinerungen von Skizzen definiert werden. Der AugIR stellt hierfür zwei verschiedene Ansätze zur Verfügung:

Erstens können Stakeholder Verfeinerungen direkt innerhalb einer geöffneten Skizze hinzufügen, indem sie an beliebiger Stelle ein Verfeinerungssymbol platzieren. In Anlehnung an die UML [68] wird hierfür ein nach unten gerichteter stilisierter Dreizack verwendet, wie er in Abbildung 3.18 dargestellt ist. Dies soll bei Stakeholdern, die mit gängigen Modellierungssprachen vertraut sind, einen intuitiven Wiedererkennungseffekt erzielen. Hierbei kann ausgewählt werden, ob zur Verfeinerung eine existierende Skizze verwendet oder eine neue erzeugt werden soll.

Zweitens kann darüber hinaus alternativ eine Verfeinerung zwischen bereits existierenden Skizzen definiert werden, indem im Navigationsgraph eine Linie vom Elternzum Kindknoten gezeichnet wird.

In beiden Fällen wird der verfeinernden Skizze automatisch ebenfalls ein entsprechendes Verfeinerungssymbol hinzugefügt, um zu signalisieren, dass es sich um die Verfeinerung einer anderen Skizze handelt. Um Verwechselungen zwischen Eltern- und Kindknoten auszuschließen wird hierzu in der verfeinernden Skizze ein um 180 ° gedrehter Dreizack verwendet, der nach oben zeigt.

Im Navigationsgraph wird die Verfeinerung durch einen gerichteten Pfeil dargestellt, der die betreffenden Skizzen miteinander verbindet, wie es in Abbildung 3.17 illustriert ist.

Die zuvor beschriebenen Möglichkeiten zur Definition von Verfeinerungen erscheinen auch für nicht-technische Stakeholder leicht verständlich und intuitiv ausführ-

bar. Weil sich Skizzen sowohl durch bereits existierende als auch durch ad hoc neu erstellte Skizzen verfeinern lassen, können Verfeinerungen spontan erfolgen und müssen nicht a priori geplant werden. Hierdurch bleibt der Modellierungsprozess informell und spontan und die Verfeinerung von Skizzen unterbricht nicht den Gedanken- und Arbeitsfluss der Stakeholder.

Die vertikale Navigation zwischen Skizzen kann über zwei verschiedene Wege erfolgen:

Zum einen können Stakeholder den Navigationsgraph verwenden, um direkt zu einer bestimmten Skizze zu gelangen. Indem sie das Vorschaubild einer Skizze in einem Skizzenknoten antippen, wird diese auf dem aktuellen Whiteboard im Vollbildmodus zur Bearbeitung geöffnet. Der Navigationsgraph kann hierbei auf einem anderen Whiteboard sichtbar bleiben, um gleichzeitig den Kontext der ausgewählten Skizze anzuzeigen.

Zum anderen können Stakeholder das dargestellte Abstraktionsniveau direkt aus einer Skizze heraus anpassen, indem sie das entsprechende Verfeinerungssymbol (vgl. Abbildung 3.18) antippen. Auf diese Weise navigieren sie vertikal zwischen den verknüpften Skizzen und ändern dabei das Abstraktionsniveau der dargestellten Inhalte.

Studien haben gezeigt, dass Softwareingenieure und Designer häufig simultan auf verschiedenen Abstraktionsebenen arbeiten und dabei selektiv den Detailgrad dargestellter Inhalte an den aktuellen Problemkontext anpassen müssen [91]. Die gleichzeitige Darstellung verschiedener Abstraktionsebenen ist dabei insbesondere bei der Bearbeitung komplexer Aufgaben hilfreich [43].

Deshalb öffnet das Antippen eines Verfeinerungssymbols ein Overlay, wie es in Abbildung 3.19 skizziert ist. Das Overlay schwebt über der aktuell geöffneten Skizze und zeigt die Inhalte der Verfeinerung an. Es können beliebig viele Overlays gleichzeitig geöffnet werden, was insbesondere dann sinnvoll ist, wenn eine Skizze mehrere Verfeinerungen enthält, die gleichzeitig dargestellt werden sollen. Die Overlays können frei auf dem Whiteboard verschoben und in ihrer Größe anpasst werden, was

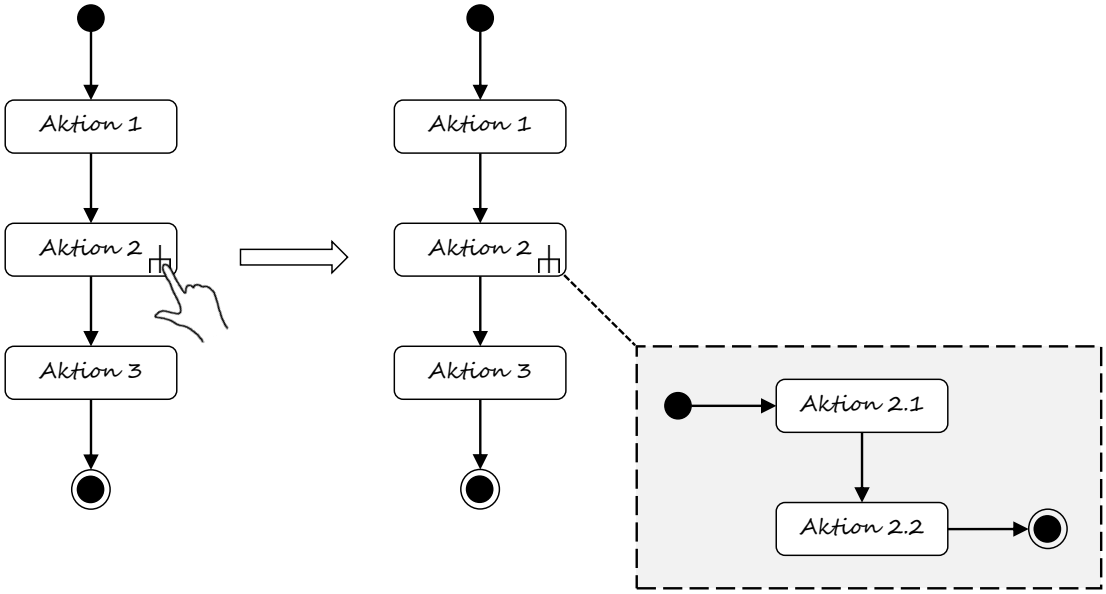

Abbildung 3.19.: Die Existenz einer Verfeinerung wird durch einen stilisierten Dreizack angedeutet. Durch Antippen des Symbols öffnet sich ein Overlay, das den Inhalt der verfeinernden Skizze anzeigt.

eine parallele Visualisierung von Verfeinerungen und ihrem umgebenden Kontext ermöglicht.

Durch Antippen eines Overlays können Stakeholder direkt zu der verfeinernden Skizze navigieren. Dies entfernt den umgebenden Kontext und öffnet die ausgewählte Skizze zur Bearbeitung im Vollbildmodus. Abbildung 3.20 illustriert dies.

Abbildung 3.21 zeigt den Ablauf in umgekehrter Richtung: Wie zuvor beschrieben deutet ein horizontal gespiegeltes Verfeinerungssymbol an, dass es sich bei der aktuellen Skizze um die Verfeinerung einer anderen Skizze handelt. Durch Antippen dieses Symbols kann erneut ein Overlay geöffnet werden, welches den Kontext der aktuellen Skizze, also den Inhalt der verfeinerten Skizze, darstellt. Durch Antippen dieses Overlays kann das Abstraktionsniveau erneut verschoben und die Details der Verfeinerung ausgeblendet werden. Stakeholder können auf diese Weise schnell und einfach zwischen verschiedenen Abstraktionsniveaus wechseln und die entsprechenden Skizzen zur Bearbeitung öffnen, ohne dass sie hierfür zum Navigationsgraph

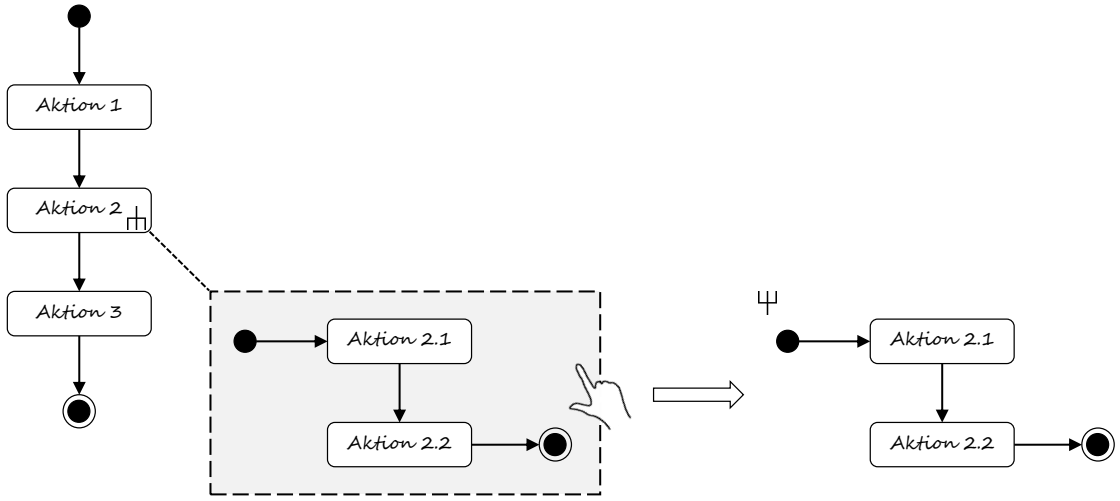

Abbildung 3.20.: Durch Antippen eines Overlays wird die zugehörige Skizze zur Bearbeitung geöffnet und der umgebende Kontext ausgeblendet.

zurückkehren müssen.

Insbesondere ist es auch möglich, Overlays für verfeinerte und verfeinernde Skizzen simultan anzuzeigen, indem die entsprechenden Verfeinerungssymbole in einer Skizze nacheinander angetippt werden. Hierdurch kann der AugIR bis zu drei Abstraktionsebenen auf einem Whiteboard gleichzeitig darstellen: Erstens die aktuell in Bearbeitung befindliche Skizze, zweitens beliebig viele Overlays für Ausschnitte aus möglichen Elternskizzen sowie drittens beliebig viele Overlays für Ausschnitte aus möglichen Kindskizzen.

3.11.2. Horizontale Navigation

In Abschnitt 3.8 wurde ein Algorithmus zur automatischen Erfassung von Trace Links präsentiert. Im Folgenden wird erläutert, wie diese Trace Links in den Skizzen visualisiert und zur horizontalen Navigation zwischen verknüpften Elementen verwendet werden können.

Eine angemessene Visualisierung von Trace Links ist grundsätzlich wichtig und in vielen Fällen nicht trivial [161]. Der AugIR verfolgt hierbei den Ansatz, erfasste Trace

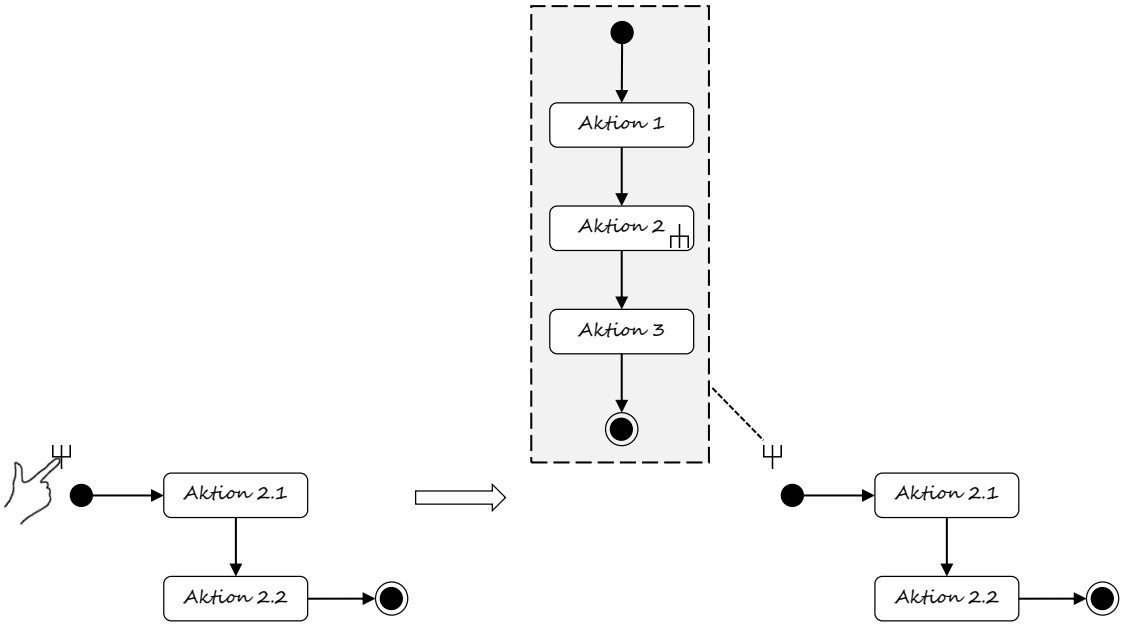

Abbildung 3.21.: Ein nach oben zeigender stilisierter Dreizack deutet an, dass es sich bei der aktuellen Skizze um die Verfeinerung einer anderen Skizze handelt. Durch Antippen des Symbls öffnet sich ein Overlay, das den Kontext der aktuellen Skizze darstellt.

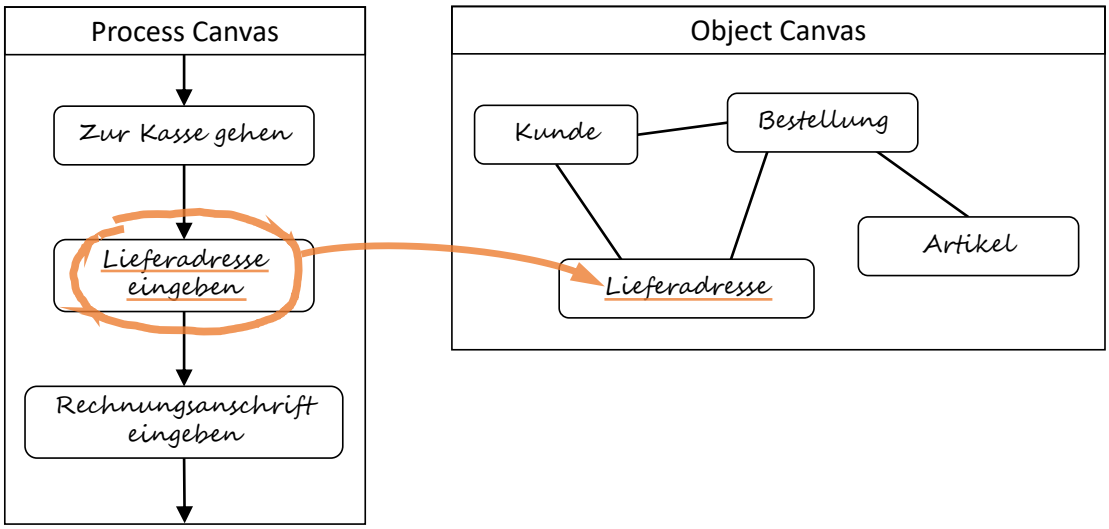

Abbildung 3.22.: Trace Links werden durch Verwendung einer Hyperlink-Metapher visualisiert, indem die verknüpften Elemente farblich unterstrichen werden.

Links möglichst intuitiv darzustellen, um insbesondere auch nicht-technischen Stakeholdern eine zugängliche Verwendung zu ermöglichen. Die Visualisierung von Trace Links in Freihandskizzen ist deshalb an das wohlbekannte Prinzip von Hyperlinks angelehnt, wie sie beispielsweise auf Webseiten und in vielen Office-Dokumenten vorkommen. Elemente einer Skizze zwischen denen ein Trace Link existiert, sind wie in Abbildung 3.22 dargestellt unterstrichen.

Eine solche Darstellung von Trace Links in Form von Hyperlinks, die sich direkt unter den betreffenden Elementen befinden, hat große Vorteile:

Stakeholder müssen nicht separate und oftmals lange Listen von Trace Links durchsuchen, um potentiell relevante Verknüpfungen aufzudecken. Stattdessen sehen sie bei ihrer Arbeit nur Trace Links für diejenigen Elemente, die zum aktuellen Zeitpunkt auf dem Whiteboard sichtbar sind und mit denen sie tatsächlich arbeiten. Hierdurch steigt die Wahrscheinlichkeit, dass die dargestellten Trace Links für den aktuellen Problemkontext wirklich relevant sind.

Darüber hinaus ist die Bedeutung der Unterstreichungen auch ohne technisches Hin-

tergrundwissen leicht verständlich, weil sie auf einem allgemein bekannten Konzept beruhen. Hierdurch können Trace Links nahtlos in den Arbeitsablauf der Stakeholder integriert werden, ohne dass ihre Verwendung zusätzliche kognitive Leistungen erfordert.

Weiterhin hilft dieser Ansatz Stakeholdern nicht nur dabei, inhaltliche Zusammenhänge aufzudecken und zu erkennen, die ansonsten möglicherweise verborgen geblieben wären. Er unterstützt sie darüber hinaus auch bei der horizontalen Navigation zwischen verknüpften Elementen, ohne dass sie hierfür den genauen Namen einer Skizze, ihren Speicherort im Projekt oder ihre Position im Navigationsgraph kennen müssen. Stakeholder können hierzu jedes unterstrichene Element in einer Skizze doppelt antippen, um eine Auflistung aller mit diesem Element verknüpften Skizzen zu erhalten. Wie in Abbildung 3.23 dargestellt, wird den Stakeholdern hierzu eine Liste mit Vorschaubildern aller verbundenen Skizzen präsentiert. Das Element, zu dem ein Trace Link existiert, wird zentriert in der Vorschau dargestellt und ist ebenfalls unterstrichen. Hierdurch wird ein unmittelbarer Hinweis auf den existierenden Zusammenhang und die Verwendung des entsprechenden Elements innerhalb der anderen Skizze gegeben.

Durch eine Wischgeste nach links oder rechts können die Vorschaubilder durchgeschaltet und der Reihe nach betrachtet werden. Die Liste ist hierbei nach Relevanz sortiert: Die Skizzen mit dem stärksten inhaltlichen Zusammenhang (mit der höchsten Kosinus-Ähnlichkeit zum betrachteten Element) stehen am Anfang der Liste, während die Skizzen mit dem niedrigsten inhaltlichen Zusammenhang (mit der geringsten Kosinus-Ähnlichkeit zum betrachteten Element) am Ende stehen. Auf diese Weise können Stakeholder den Inhalt aller verknüpften Skizzen inspizieren, ohne diese tatsächlich öffnen zu müssen.

Um zu einer bestimmten Skizze zu navigieren, kann das entsprechende Vorschaubild angetippt werden. Wie in Abbildung 3.24 dargestellt wird hierdurch die ausgewählte Skizze zur Bearbeitung im Vollbildmodus auf dem aktuellen Whiteboard geöffnet. Die Ansicht wird dabei automatisch um das Element zentriert, zu dem der Trace Link existiert.

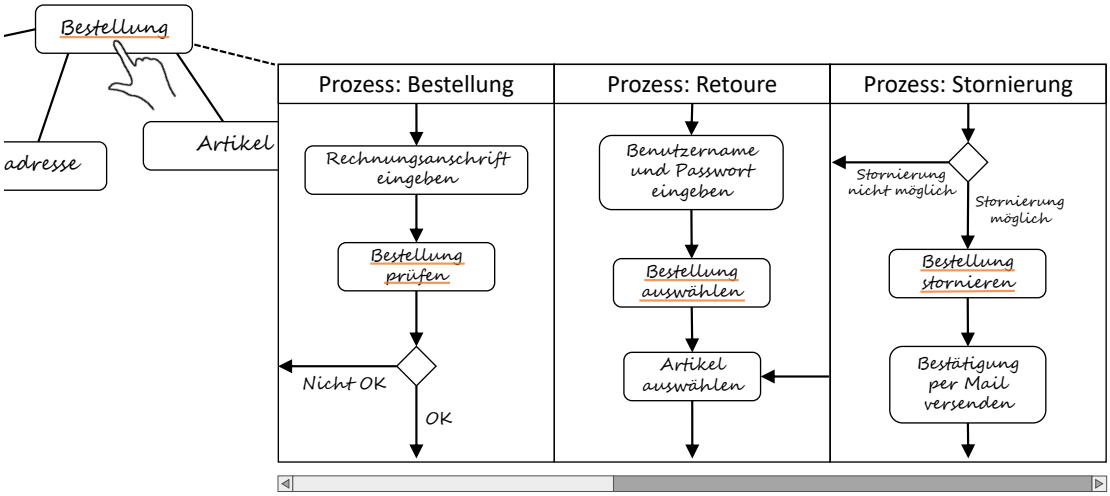

Abbildung 3.23.: Doppeltes Antippen eines Trace Links öffnet eine nach Relevanz sortierte Liste von Vorschaubildern aller verknüpften Skizzen. Das Element, zu dem ein Trace Link existiert, wird unterstrichen und zentriert dargestellt.

Um die Idee der Hyperlink-Metapher konsequent fortzuführen, werden unterhalb jeder Skizze zwei Schaltflächen angezeigt, mit denen Stakeholder vorwärts und rückwärts durch die Historie aller zuvor besuchten Skizzen navigieren können. Diese entsprechen den wohlbekannten Vor- und Zurück-Buttons eines Internet-Browsers. Durch Antippen der linken Schaltfläche können Stakeholder direkt zur Ausgangsposition springen, von der aus sie dem Trace Link gefolgt sind. Durch Antippen der rechten Schaltfläche können sie hingegen vorwärts durch die Historie der besuchten Skizzen navigieren. Hierdurch wird es Stakeholdern ermöglicht, intuitiv zwischen Skizzen zu wechseln, Inhalte zu inspizieren und auf Wunsch schnell zur ursprünglichen Position zurückzukehren.

Das vorgestellte Hyperlink-Konzept ermöglicht somit nicht nur eine Aufdeckung inhaltlicher Zusammenhänge durch intuitive Visualisierung sondern unterstützt Stakeholder darüber hinaus auch bei der horizontalen Navigation zwischen verbundenen Skizzen.

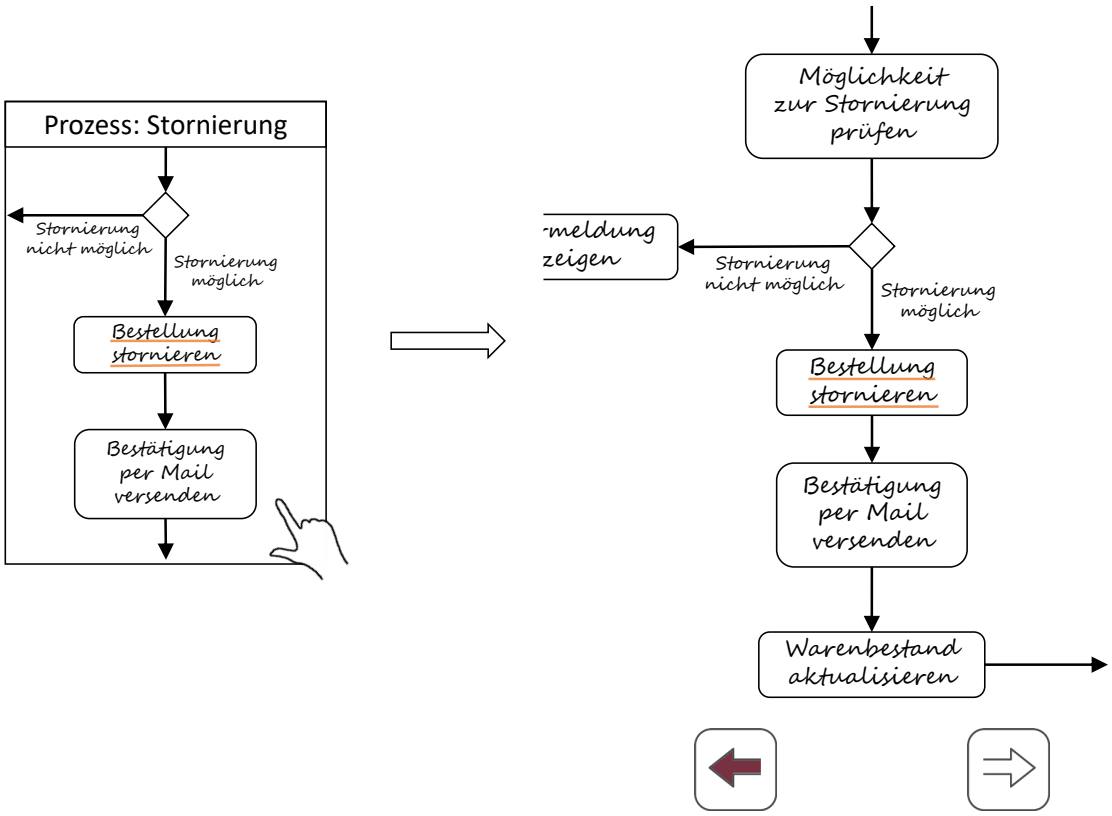

Abbildung 3.24.: Durch Antippen eines Vorschaubildes kann direkt zur ausgewählten Skizze navigiert werden. Die Ansicht wird hierbei automatisch um das unterstrichene Element zentriert, zu dem der Trace Link existiert. Unterhalb jeder Skizze befinden sich Schaltflächen, mit denen Stakeholder vorwärts und rückwärts durch die Historie aller zuvor besuchten Skizzen navigieren können.

3.12. Impact Analysis

Trace Links können Stakeholder nicht nur bei der horizontalen Navigation unterstützen. Sie können ihnen überdies auch bei der Sicherstellung der Konsistenz sowie der Identifikation von potentiellen Widersprüchen und Projektrisiken helfen.

Hierzu wird eine kontinuierliche Impact Analysis (vgl. Abschnitt 2.4) für die erfassten Trace Links durchgeführt. Dabei werden die Eingaben und Interaktionen der Stakeholder auf den elektronischen Whiteboards überwacht und deren Auswirkungen auf die miteinander verknüpften Elemente analysiert. Dies umfasst beispielsweise eine Aktualisierung der Trace Links, wenn eine existierende Linienzuggruppe gelöscht oder verändert wird.

Zum einen könnte hierdurch die textuelle Konsistenz der Skizzen sichergestellt werden: Wenn eine Verknüpfung zwischen zwei Elementen beispielsweise auf einem gemeinsamen Schlüsselwort w basiert und Stakeholder dieses Wort an einer Stelle ändern oder löschen, kann dies dazu führen, dass der entsprechende Trace Link obsolet wird und entfernt werden muss. Stakeholder könnten auf diesen Umstand hingewiesen und es könnte ihnen vorschlagen werden, das entsprechende Wort w an den vormals zuvor verknüpften Stellen zu betrachten und auch dort gegebenenfalls anzupassen oder zu entfernen, um die inhaltliche Konsistenz zu wahren. Um den Gedanken- und Arbeitsfluss der Stakeholder während eines Interaction-Room-Workshops nicht zu stören, kommt eine solche Technik im AugIR jedoch nicht zum Einsatz.

Vielmehr unterstützt der AugIR die Interaction-Room-Methode, indem er sämtliche Annotationsaktivitäten überwacht und damit die Konsistenz der platzierten Annotationen sicherstellt sowie Stakeholder auf potentielle Widersprüche und Projektrisiken hinweist. Dies wird in den folgenden Abschnitten näher erläutert.

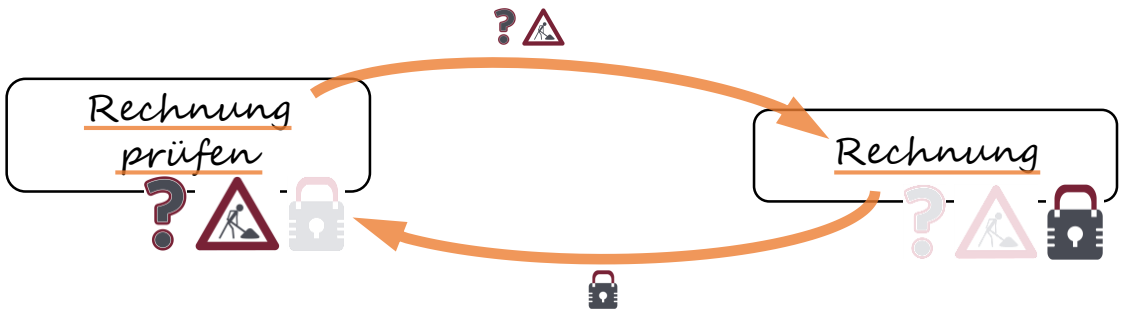

Abbildung 3.25.: Indirekte Annotationen werden von dem Element, auf dem sie platziert wurden, automatisch über Trace Links auf verknüpfte Elemente übertragen. Sie werden heller dargestellt als gewöhnliche Annotationen, um sie von diesen abzugrenzen.

3.12.1. Indirekte Annotationen

Immer wenn eine Linienzuggruppe mit einer Annotation versehen wird, ermittelt der AugIR sämtliche Elemente, die mit dieser Gruppe über Trace Links verbunden sind. Die platzierte Annotation wird daraufhin automatisch auch auf die verknüpften Elemente angewendet, wie es in Abbildung 3.25 dargestellt ist. Diese Annotationen, die über Trace Links auf andere Elemente übertragen werden, heißen *indirekte Annotationen*. Sie werden heller dargestellt als gewöhnliche Annotationen, um sie von diesen abzugrenzen.

Indirekte Annotationen stellen automatisch die Konsistenz der Inhalte hinsichtlich der platzierten Annotationen sicher. Sie sorgen dafür, dass ein Element, welches an mehreren Stellen in unterschiedlichen Canvases vorkommt, überall mit den gleichen Annotationen versehen ist.

Weiterhin erleichtern sie die Analyse von Annotationen. Sie zeigen an, welche Annotationen insgesamt auf einem Element sowie auf allen damit verknüpften Entitäten platziert wurden, ohne dass hierfür alle Skizzen des Projekts manuell durchsucht werden müssen. Dies ist nicht nur für die Nachbereitung von Interaction-Room-Workshops nützlich – Stakeholder erhalten hierdurch bereits während eines Work-

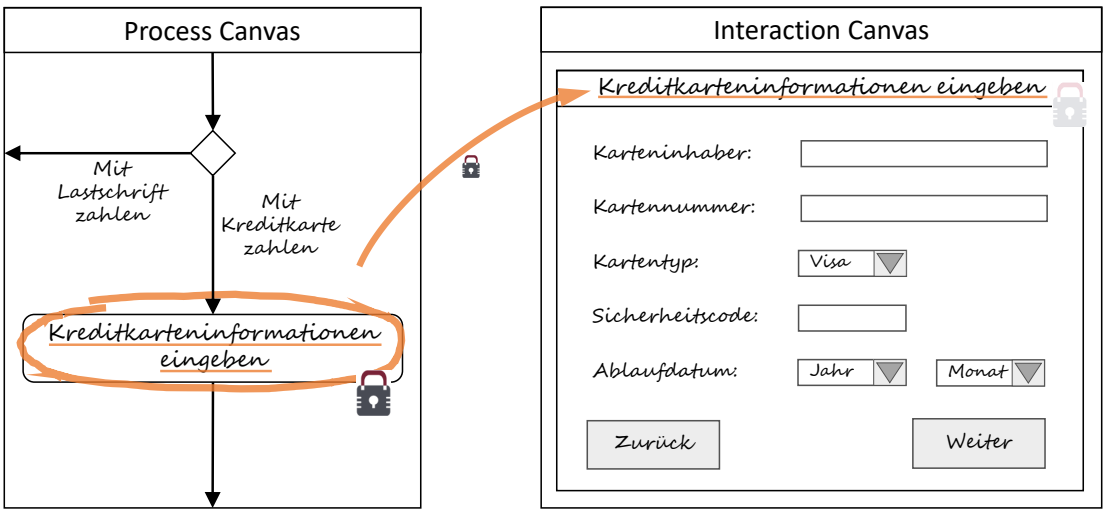

Abbildung 3.26.: Beispiel für die Nützlichkeit von indirekten Annotationen: Bei der Modellierung eines Dialogs zur Eingabe von Kreditkarteninformationen ist ersichtlich, dass der zugehörige Prozess bereits zuvor als sicherheitskritisch annotiert wurde.

shops einen globalen Überblick über alle bisher verwendeten Annotationen, die für den aktuell dargestellten Problemkontext potentiell relevant sein könnten.

Abbildung 3.26 illustriert dies an einem einfachen Beispiel: In einem früheren Workshop wurde ein Prozess zur Bestellabwicklung in einem Online-Shop diskutiert und annotiert, der eine Aktion zur Eingabe und Verarbeitung von Kreditkarteninformationen enthält. Weil es sich hierbei um einen sicherheitskritischen Ablauf handelt, wurde die entsprechende Aktion dabei von den Stakeholdern mit der Annotation „Sicherheit“ versehen. Wenn zu einem späteren Zeitpunkt die zugehörige grafische Benutzerschnittstelle skizziert wird und zwischen dieser und dem zuvor modellierten Prozess ein Trace Link existiert, so wird die Sicherheitsannotation als indirekte Annotation auf die UI-Skizze übertragen. Den Designern steht somit bei der Modellierung des UIs die entsprechende Information hinsichtlich der bereits zuvor identifizierten Sicherheitsaspekte zur Verfügung, so dass sie diese unmittelbar in ihre Entwurfsentscheidungen einfließen lassen können.

Je nach Art des Trace Links ist eine Übertragung der Annotationen jedoch nicht

in jedem Fall immer wünschenswert. Stakeholder können indirekte Annotationen deshalb manuell entfernen, wenn sie für ein verbundenes Element nicht zutreffend sind. Dies hat keine Auswirkungen auf das ursprünglich annotierte Element.

Darüber hinaus können indirekte Annotationen zu jeder Zeit komplett ein- und ausgeblendet werden, um die Darstellung der Inhalte an die aktuellen Erfordernisse der Stakeholder anzupassen. Dies ist beispielsweise sinnvoll, um Stakeholder beim Annotieren nicht durch Annotationen zu beeinflussen, die von anderen Teilnehmern zuvor in anderen Skizzen platziert wurden.

Indirekte Annotationen können außerdem bei der Aufdeckung möglicher Widersprüche unterstützen. Wenn ein Element beispielsweise gleichzeitig mit den Annotationen „Unveränderlichkeit“ und „Verbesserungsbedarf“ gekennzeichnet ist, so impliziert dies einen potentiellen Widerspruch, weil eine gleichzeitige Bewahrung und Veränderung des Systems sich in der Regel gegenseitig ausschließen.

Wenn die Annotationen auf dem selben Element platziert wurden, können solche Widersprüche leicht erkannt werden. Kommt das gleiche Element jedoch in unterschiedlichen Canvases vor und wird dort mit entsprechenden Annotationen versehen, so ist eine Identifikation der hierdurch entstehenden Widersprüche schwierig.

Indirekte Annotationen helfen hierbei, weil sie leicht automatisch ausgewertet werden können. Hierzu werden sowohl die indirekten als auch die direkt platzierten Annotationen für jedes Element betrachtet und hinsichtlich potentieller Widersprüche überprüft, die in der Interaction-Room-Methode [29] aufgeführt sind. Falls ein möglicher Widerspruch vorliegt, kann ein entsprechender Warnhinweis präsentiert werden. Dies sollte jedoch nicht während eines Workshops sondern erst im Anschluss daran im Rahmen einer dedizierten Analyse erfolgen, um den Gedanken- und Arbeitsfluss der Stakeholder nicht zu stören.

3.12.2. Annotationsaggregationen

Aufbauend auf den zuvor eingeführten indirekten Annotationen können Annotationen darüber hinaus transitiv über Trace Links zu einer gemeinsamen Liste zusam-

mengefasst werden, um Stakeholder bei der Identifikation potentieller Projektrisiken zu unterstützen. Dieser Vorgang wird im Folgenden als *Annotationsaggregation* bezeichnet.

Die Grundidee hierbei ist Folgende: Typische Interaction-Room-Workshops starten zumeist mit der Erstellung eines Feature Canvas. Danach werden relevante Prozesse auf den Process Canvases skizziert, wichtige Fachobjekte auf dem Object Canvas gesammelt und abschließend eventuelle Schnittstellen auf dem Integration Canvas betrachtet. Man überlegt sich, dass diese Canvases zusammenhängen und über Trace Links miteinander verbunden sein können: Die auf Karteikarten notierten Features des Feature Canvas werden durch Aktionen in den skizzierten Prozessen realisiert. Diese Aktionen verarbeiten Objekte des Object Canvas, welche wiederum über die Schnittstellen, die auf dem Integration Canvas modelliert sind, ausgetauscht werden.

Anstatt wie bei indirekten Annotationen die Annotationen nur auf das unmittelbar verknüpfte Element zu übertragen, werden bei der Annotationsaggregation alle Annotationen aller miteinander verbundenen Elemente gesammelt und wenn möglich unter einem passendem Element wie beispielsweise einem Feature aggregiert. Aus Art und Häufigkeit der verwendeten Annotationen lassen sich hierdurch Rückschlüsse auf potentielle Projektrisiken ziehen, die mit der Umsetzung des jeweiligen Elements verbunden sind.

Abbildung 3.27 illustriert diesen Prozess anhand eines simplen Beispiels: Im Rahmen eines Workshops wurden Feature, Process, Object und Integration Canvases erstellt. Durch Anwendung der Annotationsaggregation wurden fünf Annotationen vom Typ „Ungewissheit“ zu einem betrachteten Feature aggregiert. Dies deutet bereits an, dass hinsichtlich dieses Features erhöhte Unsicherheit unter den Stakeholdern herrscht, die insbesondere von der rechten Aktion und den damit verbundenen Objekten und Schnittstellen ausgeht. Darüber hinaus wurden ebenfalls drei Annotationen vom Typ „Sicherheit“ aggregiert. Dies deutet auf eine zusätzliche Gefahrenquelle hin, weil sicherheitsrelevante Aspekte eines Systems üblicherweise besonders gut durchdacht und von den Stakeholdern verstanden sein sollten.

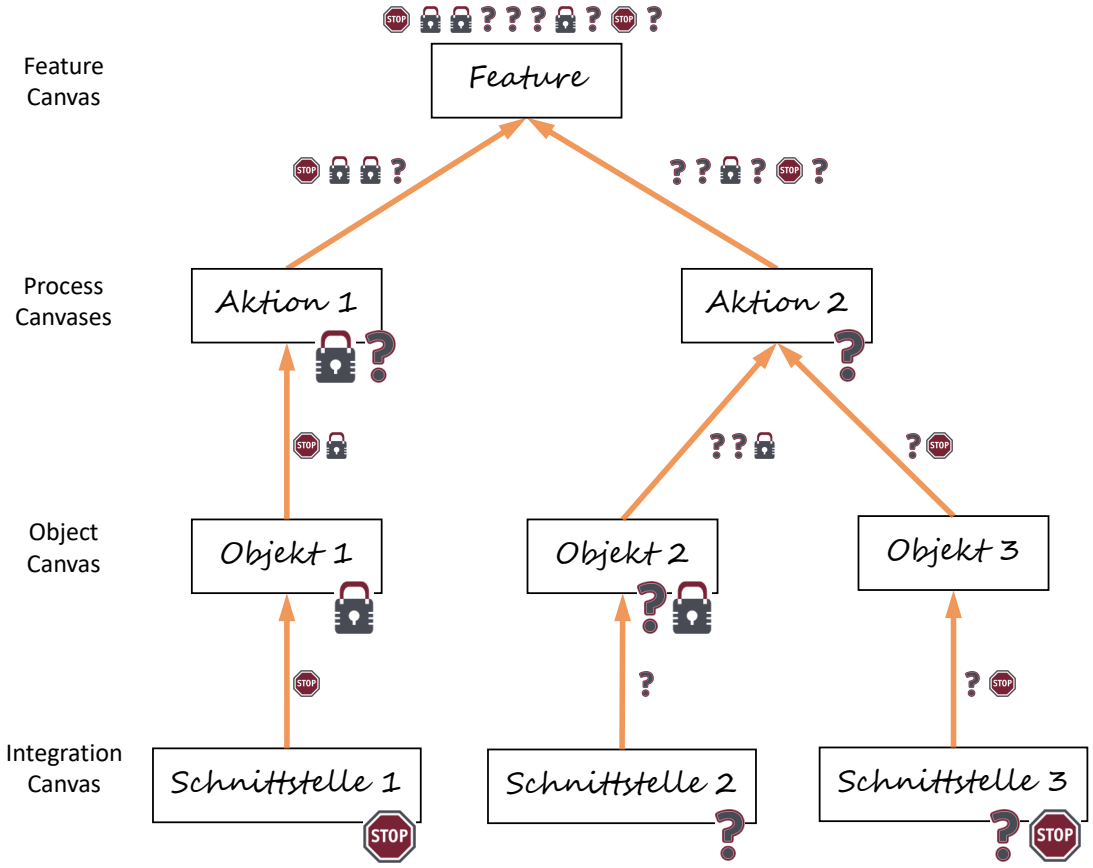

Abbildung 3.27.: Trace Links ermöglichen die automatische Aggregation von Annotationen, die auf zusammenhängenden Elementen platziert wurden.

Annotationsaggregationen können somit insbesondere im Rahmen der Nachbereitung eines Interaction-Room-Workshops bei der Identifikation potentieller Projektrisiken helfen, die ansonsten möglicherweise unentdeckt geblieben wären oder nur mit großem manuellem Aufwand hätten erkannt werden können.

3.13. Zusammenfassung

In diesem Kapitel wurde ausführlich das Konzept des Augmentierten Interaction Rooms zur Lösung der in Abschnitt 1.3 diskutierten Probleme präsentiert.

Zunächst wurde beschrieben, wie mit Hilfe der logistischen Regression Linienzüge als Text oder Form klassifiziert werden können. Hierzu werden sowohl lokale als auch globale Features von Linienzügen betrachtet. Während sich lokale Features nur aus den geometrischen Eigenschaften eines einzelnen Linienzugs wie beispielsweise seiner Länge und der Anzahl seiner Kontrollpunkte bestimmen, berücksichtigen globale Features die räumliche und zeitliche Nähe eines Linienzugs zu seinen Nachbarn. Insgesamt wurden 24 relevante Features zur Klassifizierung von Linienzügen identifiziert.

Um auf den potentiellen Text-Linienzügen eine Handschrifterkennung durchführen zu können, werden diese automatisch zu Linienzuggruppen zusammengefasst. Hierbei kommt ein Algorithmus zum Einsatz, der für jeden gezeichneten Linienzug den zeitlichen und räumlichen Abstand zu bereits existierenden Elementen auf der Zeichenfläche betrachtet und in Abhängigkeit davon eine passende Gruppe bestimmt. In Übereinstimmung mit verwandten Ansätzen wird eine maximale zeitliche Distanz von 3,5 Sekunden und ein maximaler euklidischer Abstand von 10 Pixeln für benachbarte Linienzüge gewählt.

Nach der Durchführung einer Handschrifterkennung auf den gruppierten Linienzügen werden die erkannten Texte durch Anwendung von Tokenisierungsregeln in Tupel zerlegt, um sie für eine weiterführende Textanalyse vorzubereiten. Hierbei kommen drei heuristische Regeln t_1, t_2 und t_3 zum Einsatz, die aus der Analyse von beobachteten Problemen bei der Handschrifterkennung abgeleitet wurden.

Um darüber hinaus die Korrelation potentiell zusammenhängender Elemente zu stärken und damit die Erfassung von Trace Links zu verbessern, werden die erzeugten Tokenisierungen in einem nachfolgenden Schritt analysiert und falls nötig modifiziert. Hierbei wird für jedes Wort einer Tokenisierung geprüft, ob dieses in einer anderen betrachteten Skizze vorkommt. Falls dies nicht der Fall ist, wird das betreffende Element durch ein ähnliches Wort aus der betrachteten Skizze ersetzt, sofern die Ähnlichkeit zu diesem hoch genug ist. Im Rahmen eines Experiments wurde ermittelt, dass $\mu := 0{,}75$ ein geeigneter Schwellwert für diese Ähnlichkeitsbetrachtung ist.

Auf den finalen Tokenisierungen wird ein Algorithmus zur Erfassung von Trace Links angewendet, der das Vector Space Model nutzt. Hierbei werden Häufigkeitsvektoren für die Tokenisierungen der Texte aller Linienzuggruppen berechnet und mit Hilfe der Kosinus-Ähnlichkeit miteinander verglichen. Liegt die Ähnlichkeit über einem zuvor definierten Schwellwert ν, so wird zwischen den betreffenden Elementen ein Trace Link erfasst. Ein geeigneter Wert für ν muss noch festgelegt werden. Dies erfolgt in Kapitel 7 im Rahmen eines umfangreichen Experiments zur Bewertung der erreichbaren Qualität bei der Trace-Link-Erfassung.

Nach der Diskussion der notwendigen Analyse- und Vorverarbeitungsschritte wurde in diesem Kapitel außerdem erläutert, wie Skizzen im AugIR erstellt und verwaltet werden können. Zur Interaktion mit den digitalen Inhalten kommen dabei drei verschiedene Werkzeuge zum Einsatz: Stakeholder können mit speziellen Stiften auf die Fläche des elektronischen Whiteboards schreiben und Freihandskizzen erstellen, sie können mit einem speziellen Radierschwamm Inhalte pixelgenau ausradieren und überdies mit ihren Fingern die skizzierten Inhalte manipulieren und die Ansicht beeinflussen.

Während das vorgestellte Konzept bereits eine problemlose Digitalisierung generischer Skizzen und der meisten Interaction-Room-Canvases ermöglicht, müssen Feature und Interaction Canvas hinsichtlich der zugrundeliegenden Problemstellung gesondert betrachtet werden:

Zur Abbildung des Features Canvas im AugIR kommen mobile Endgeräte zum Einsatz, welche die Erstellung digitaler Feature-Karten ermöglichen. Auf diese Weise

können Medienbrüche vermieden werden, indem Stakeholder digitale Karteikarten auf ihren individuellen Tablets erstellen und drahtlos auf die elektronischen Whiteboards übertragen.

Der Interaction Canvas wird durch ein digitales Storyboard realisiert, das die Verknüpfung von UI-Skizzen über benutzerdefinierte Gesten und deren interaktive Ausführung in einem Simulationsmodus ermöglicht. Dies erlaubt eine adäquate Verhaltensbeschreibung von Benutzerschnittstellen sowie eine Unterstützung der Diskussion von Interaktionsmöglichkeiten. Zur Notation von Gesten werden die von Hesenius et al. [103, 104] vorgeschlagenen GestureCards verwendet. Hierbei handelt es sich um eine hybride Repräsentation von Gesten, die grafische und textuelle Elemente miteinander verbindet.

Zur Verwaltung der erstellten Skizzen und Verknüpfungen kommt im AugIR eine Graph-Datenstruktur zum Einsatz, die als Navigationsgraph bezeichnet wird. Dieser erlaubt eine hierarchische Strukturierung der Inhalte durch Gruppen- und Skizzenknoten, die über Verschachtelungen, Gruppierungen, Verfeinerungen und Trace Links miteinander verbunden werden können. Auf diese Weise kann der Navigationsgraph übersichtlich mehrere Abstraktionsebenen abbilden und unterstützt überdies die simultane Visualisierung von mikroskopischen und makroskopischen Inhalten des betrachteten Softwareprojekts.

Verfeinerungen von Skizzen ermöglichen hierbei die einfache und schnelle vertikale Navigation zwischen diesen. Dazu wird eine Technik verwendet, welche die Inhalte einer Verfeinerung oder den Kontext einer betrachteten Skizze in einem Overlay anzeigt. Dies ermöglicht die gleichzeitige Darstellung von bis zu drei Abstraktionsebenen auf dem selben Whiteboard.

Durch die erfassten Trace Links kann zudem die horizontale Navigation zwischen Skizzen unterstützt werden. Trace Links werden hierzu durch Hyperlinks in den Skizzen direkt unter den Elementen visualisiert, zu denen sie gehören. Dies erlaubt die Inspektion von Inhalten, die für den aktuellen Problemkontext relevant sein könnten, ohne hierfür zunächst umständlich zur entsprechenden Skizze navigieren oder diese suchen zu müssen. Hierdurch können außerdem Zusammenhänge und Ab-

hängigkeiten aufgedeckt werden, die ohne eine geeignete Trace-Link-Visualisierung möglicherweise verborgen geblieben wären.

Durch eine kontinuierliche Impact Analysis werden darüber hinaus die Annotationsaktivitäten der Stakeholder überwacht. Dies ermöglicht die Realisierung von indirekten Annotationen, die automatisch über Trace Links auf verbundene Elemente übertragen werden. Zum einen lassen sich auf diese Weise Inkonsistenzen vermeiden, weil alle zusammenhängenden Elemente mit den gleichen Annotationen versehen sind. Zum anderen können sie zur Identifikation von potentiellen Widersprüchen automatisch analysiert werden.

Durch eine transitive Übertragung von Annotationen auf verknüpfte Elemente lassen sich darüber hinaus Annotationsaggregationen erfassen. Hierzu können Annotationen beispielsweise zu einem Feature aggregiert werden, um aus Art und Häufigkeit der verwendeten Annotationen Rückschlüsse auf potentielle Projektrisiken zu ziehen, die mit der Umsetzung dieses Features verbunden sind.

Eine abschließende Diskussion des vorgestellten Konzepts hinsichtlich der in Abschnitt 1.3 genannten Probleme erfolgt in Kapitel 9 unter Berücksichtigung der in Teil III dieser Arbeit durchgeführten Evaluationen.

4. Technische Umsetzung

Um eine Evaluation des in Kapitel 3 vorgestellten Konzepts zu ermöglichen, wurde dieses prototypisch implementiert. Abbildung 4.1 zeigt ein Foto der laufenden AugIR-Software im Labor der Universität Duisburg-Essen. Auf dem linken Whiteboard ist ein Process Canvas zur Bearbeitung geöffnet. Das rechte Whiteboard zeigt den Navigationsgraphen mit zwei Gruppen und mehreren Skizzen.

Als elektronische Whiteboards werden sogenannte *SMART Boards* [214] verwendet, die bereits vielfach mit Erfolg in verwandten Arbeiten eingesetzt wurden [50, 116, 129, 231]. Diese Boards verfügen nicht über kapazitive Displays, wie es bei vielen Multi-Touch-Oberflächen üblich ist. Stattdessen sind sie mit mehreren hochsensitiven Kameras ausgestattet, welche die Interaktionen der Benutzer mit dem Board registrieren. Über einen entsprechenden Treiber werden die erkannten Eingaben in Events übersetzen, die von der Ereignisbehandlung einer Software empfangen und verarbeitet werden können. Das Bild wird dabei über einen Kurzdistanzbeamer, der über dem Board an der Wand befestigt ist, auf die Oberfläche des SMART Boards projiziert.

SMART Boards haben gegenüber vielen anderen elektronischen Whiteboards insbesondere den entscheidenden Vorteil, dass ihre Kameras zwischen verschiedenen physischen Eingabewerkzeugen unterscheiden können. Wie in Abschnitt 3.9 beschrieben, lassen sich hierdurch verschiedene Eingabemodalitäten durch Finger, Stifte und Radierer realisieren, die jeweils unterschiedliche Aktionen auslösen. Dies spielt eine große Rolle für die intuitive Benutzung, weil Stakeholder hierdurch genau wie auf traditionellen analogen Whiteboards mit Stiften zeichnen und einem speziellen Radierschwamm radieren können.

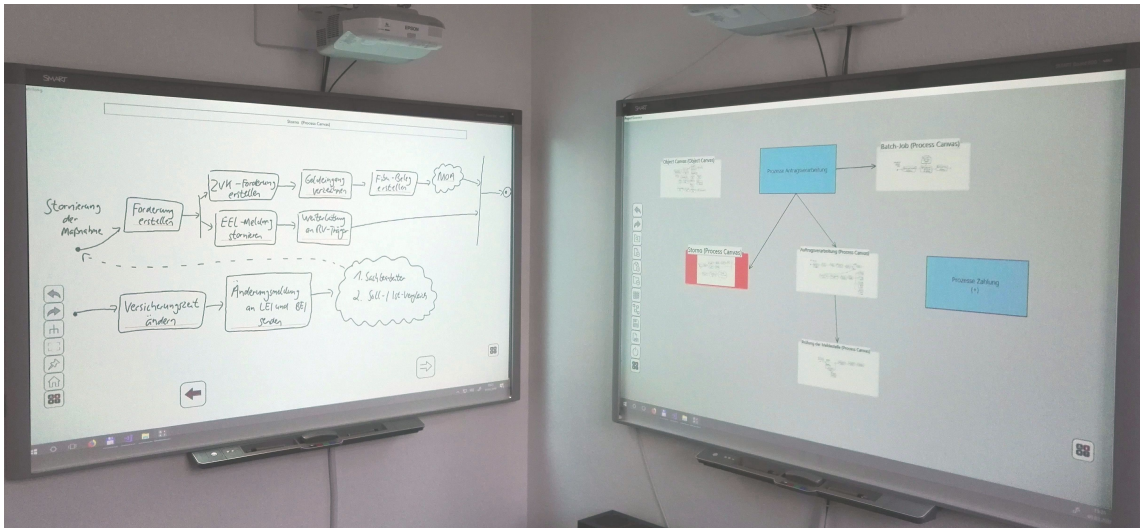

Abbildung 4.1.: Foto des AugIR-Prototyps im Universitätslabor.

Der AugIR-Prototyp wurde unter Windows in der Programmiersprache C# mit dem .NET-Framework entwickelt, weil hierfür vom Hersteller der SMART Boards das umfangreichste und stabilste Software Development Kit (SDK) zur Verfügung gestellt wird. Die Implementierung umfasst über 400 Klassen mit etwa 20.000 Zeilen Quellcode. Zur Realisierung der grafischen Benutzerschnittstelle wird die Windows Presentation Foundation (WPF) [169] verwendet. Zur Handschrifterkennung kommen die Microsoft.Ink-Bibliotheken [168] zum Einsatz.

Die folgenden Abschnitte sollen einen Einblick in Architektur, Prozesse und technische Details der Implementierung geben. Aufgrund des enormen Umfangs können hierbei jedoch nur einige ausgewählte Aspekte kurz beschrieben werden.

4.1. Ansichten und Modi

In Abbildung 4.2 sind die verschiedenen Modi illustriert, in denen sich die AugIR-Software zur Laufzeit befinden kann. Die Abbildung zeigt dabei jedoch nur die möglichen Zustände eines einzelnen elektronischen Whiteboards. Weil im AugIR in

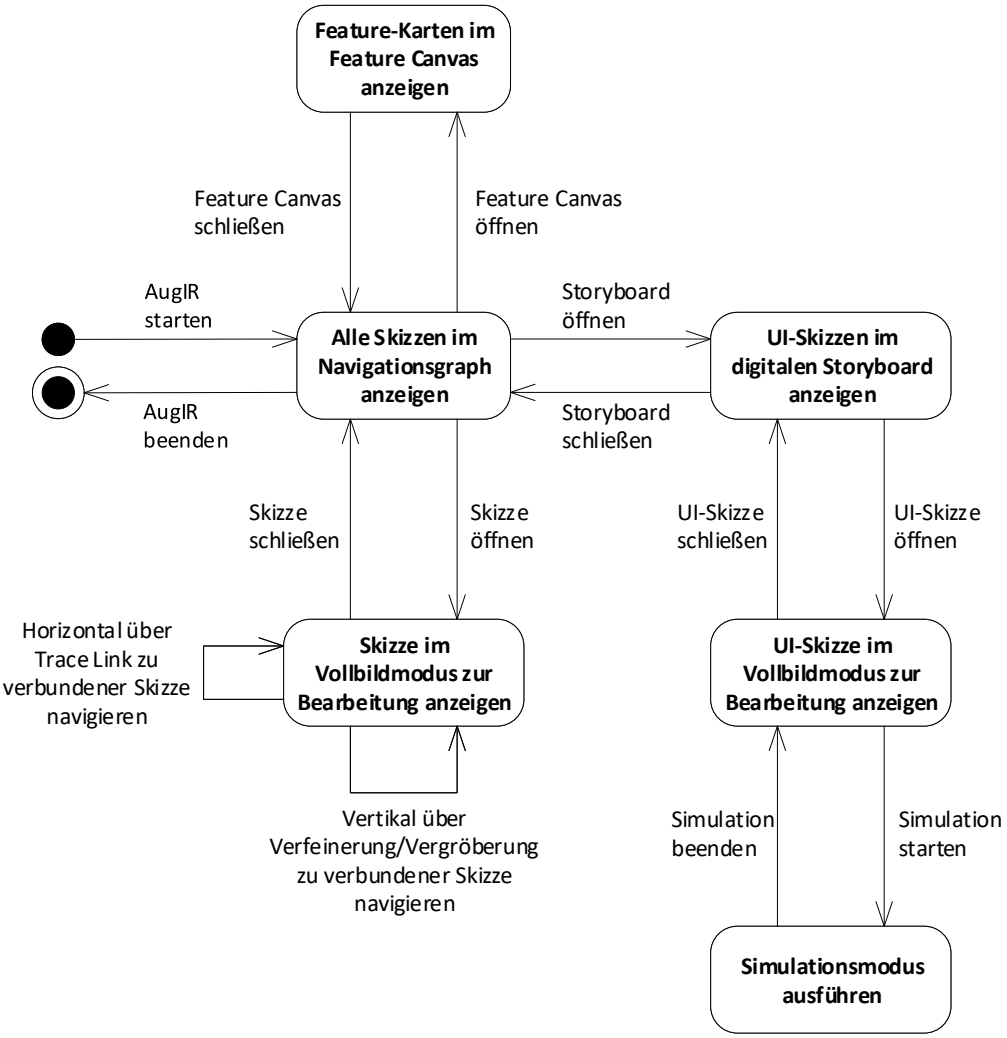

Abbildung 4.2.: UML-Zustandsdiagramm für mögliche Modi des AugIRs auf einem elektronischen Whiteboard.

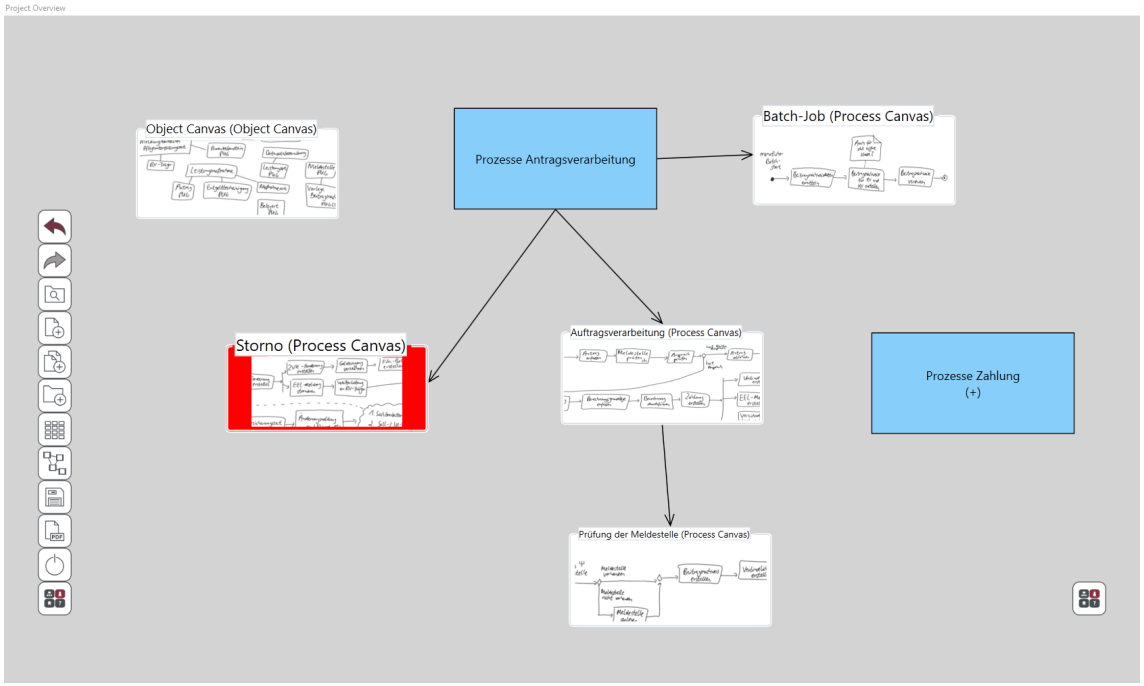

Abbildung 4.3.: Screenshot des Navigationsgraphen im AugIR-Prototyp.

der Regel mehrere miteinander verbundene Displays zum Einsatz kommen, kann sich jedes Whiteboard individuell in einem eigenen Zustand befinden.

Nach dem Start der AugIR-Software wird zunächst der in Abschnitt 3.10 beschriebene Navigationsgraph angezeigt. Dieser stellt alle Skizzen sowie eventuell definierte Gruppen und Verknüpfungen übersichtlich als Graph dar. Alle Knoten können in dieser Ansicht beliebig verschoben und frei auf dem elektronischen Whiteboard angeordnet werden, um das Projekt geeignet zu strukturieren. Abbildung 4.3 zeigt beispielhaft einen Screenshot des Navigationsgraphen im AugIR-Prototyp. Dieser enthält zwei Gruppen (blaue Knoten), von denen eine aus- und die andere eingeklappt ist. Die ausgeklappte Gruppe enthält mehrere Skizzen, wobei eine dieser Skizzen durch eine weitere verfeinert ist. Jede Skizze wird durch eine Miniaturvorschau im Navigationsgraphen visualisiert. Der linke Process Canvas ist aktuell zur Bearbeitung auf einem anderen Whiteboard im AugIR geöffnet und deshalb farblich hervorgehoben.

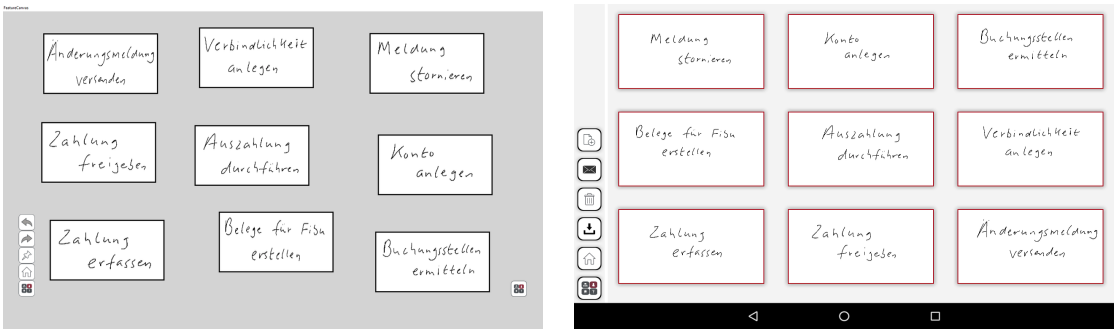

Abbildung 4.4.: Screenshot des Feature Canvas im AugIR-Prototyp (links) und der zugehörigen Android-App zur Erstellung von Feature-Karten (rechts).

Vom Navigationsgraphen aus können Stakeholder über einen Button in der linken Seitenleiste den in Abschnitt 3.9.1 beschriebenen Feature Canvas öffnen, um Feature-Karten zu betrachten, zu diskutieren und zu annotieren. Abbildung 4.4 zeigt einen Screenshot des Feature Canvas im AugIR-Prototyp sowie ein Bild der zugehörigen Android-Anwendung zur Erstellung von Feature-Karten. Die Karten können von den Stakeholdern auf ihren individuellen mobilen Endgeräten mit speziellen Stiften geschrieben und anschließend über eine WLAN-Verbindung drahtlos an das Whiteboard übertragen werden. Auf dem Feature Canvas werden die Karten als rechteckige Elemente dargestellt, die frei auf der Arbeitsfläche verschoben und beliebig angeordnet werden können. Die Inhalte werden dabei genau so dargestellt, wie sie von den Stakeholdern handschriftlich auf den Mobilgeräten verfasst wurden.

Ebenso können Stakeholder im Navigationsgraph eine bereits existierende Skizze antippen oder über das Seitenmenü eine neue Skizze erstellen, um diese zur Bearbeitung im Vollbildmodus zu öffnen. Stakeholder können daraufhin Inhalte mit speziellen Stiften und einem Radierschwamm editieren sowie mit ihren Fingern manipulieren.

Wenn in einer Skizze Trace Links erkannt wurden, so können Stakeholder durch doppeltes Antippen eines entsprechend unterstrichenen Elements wie in Abschnitt 3.11.2 beschrieben eine Vorschauansicht auf alle damit verknüpften Elemente erhalten. Dies wird in Abbildung 4.5 illustriert, die einen Ausschnitt eines Object Canvas

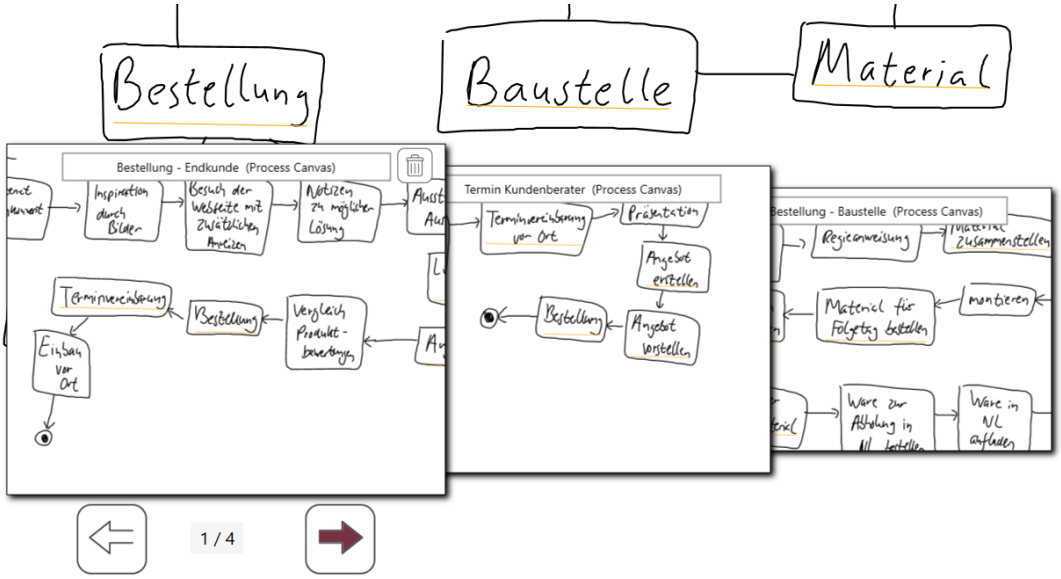

Abbildung 4.5.: Screenshot der Trace-Link-Vorschau für das Element „Bestellung“ im AugIR-Prototyp.

mit geöffneter Vorschauansicht für das Element „Bestellung“ zeigt. Dieses ist über Trace Links mit vier Elementen aus anderen Skizzen verbunden, die jeweils zentriert in den Vorschaubildern angezeigt werden. Die Bilder können mit einer Wischgeste nach links oder rechts oder durch Antippen der Pfeile darunter durchgeschaltet werden. Wenn Stakeholder einen falsch positiven Trace Link entdecken, können sie diesen über das Mülltonnen-Symbol in der rechten oberen Ecke der entsprechenden Vorschau entfernen. Durch Antippen eines Vorschaubildes können sie direkt zum jeweiligen Element in der ausgewählten Skizze navigieren.

Falls eine Skizze ein oder mehrere Verfeinerungssymbole enthält, können Stakeholder durch doppeltes Antippen dieses Symbols wie in Abschnitt 3.11.1 erläutert ein Overlay öffnen, welches die Details der entsprechenden Verfeinerung oder Vergröberung anzeigt. Abbildung 4.6 illustriert dies beispielhaft durch einen Screenshot, der das Overlay einer Verfeinerung des Elements „Meldestelle prüfen“ darstellt. Das nach unten gerichtete Dreizacksymbol deutet an, dass das hiermit versehene Element verfeinert wurde. Entsprechend zeigt das nach oben gerichtete Dreizacksymbol in der verbundenen Prozessskizze an, dass es sich um die Verfeinerung eines Elements aus

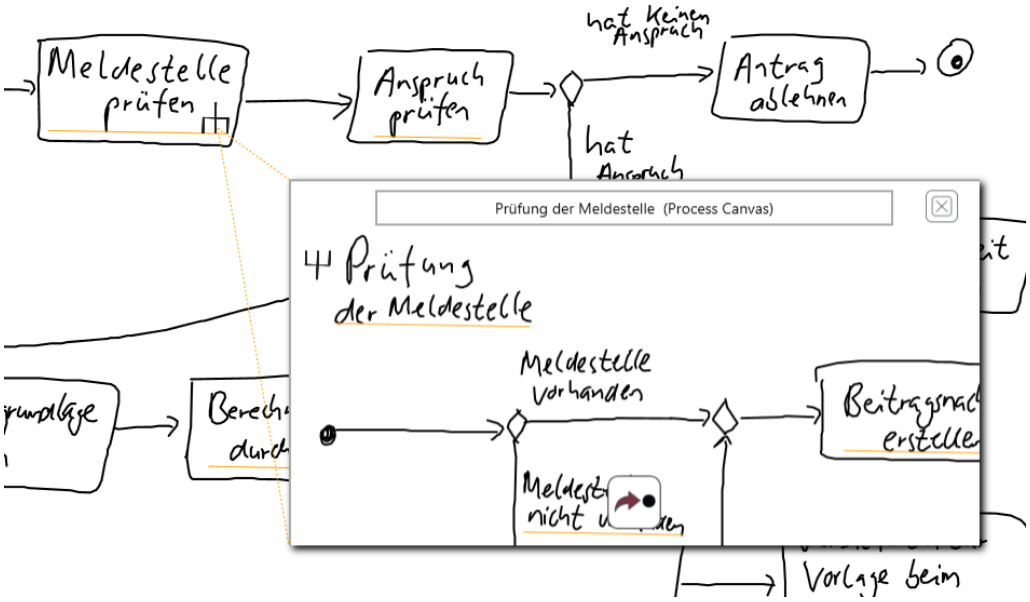

Abbildung 4.6.: Screenshot des Overlays zur detaillierten Darstellung der Verfeinerung des Elements „Meldestelle prüfen“ im AugIR-Prototyp.

einer anderen Skizze handelt. Die verbundene Skizze kann direkt in dem Overlay bearbeitet werden, ohne dass sie zuvor explizit geöffnet werden muss. Um dennoch dorthin zu navigieren, können Stakeholder den Button am unteren Rand des Overlays verwenden. Dies öffnet die entsprechende Skizze zur Bearbeitung im Vollbildmodus.

Weiterhin können Stakeholder vom Navigationsgraph aus das in Abschnitt 3.9.2 beschriebene digitale Storyboard des Interaction Canvas öffnen. Während der Navigationsgraph alle Skizzen eines Projekts anzeigt, stellt das Storyboard ausschließlich die UI-Skizzen und die Dialogübergänge zwischen diesen dar. Abbildung 4.7 zeigt beispielhaft den Screenshot eines digitalen Storyboards aus einer Fallstudie zur Modellierung eines Foto-Editors. Die UI-Skizzen können als Vorschaubilder frei auf dem Storyboard platziert und zur Definition von Dialogflüssen miteinander verbunden werden. Jede Kante ist dabei mit einer GestureCard versehen, die anzeigt, mit welcher Geste die entsprechende Transition ausgelöst werden kann.

Darüber hinaus können über das Storyboard UI-Skizzen zur Bearbeitung im Voll-

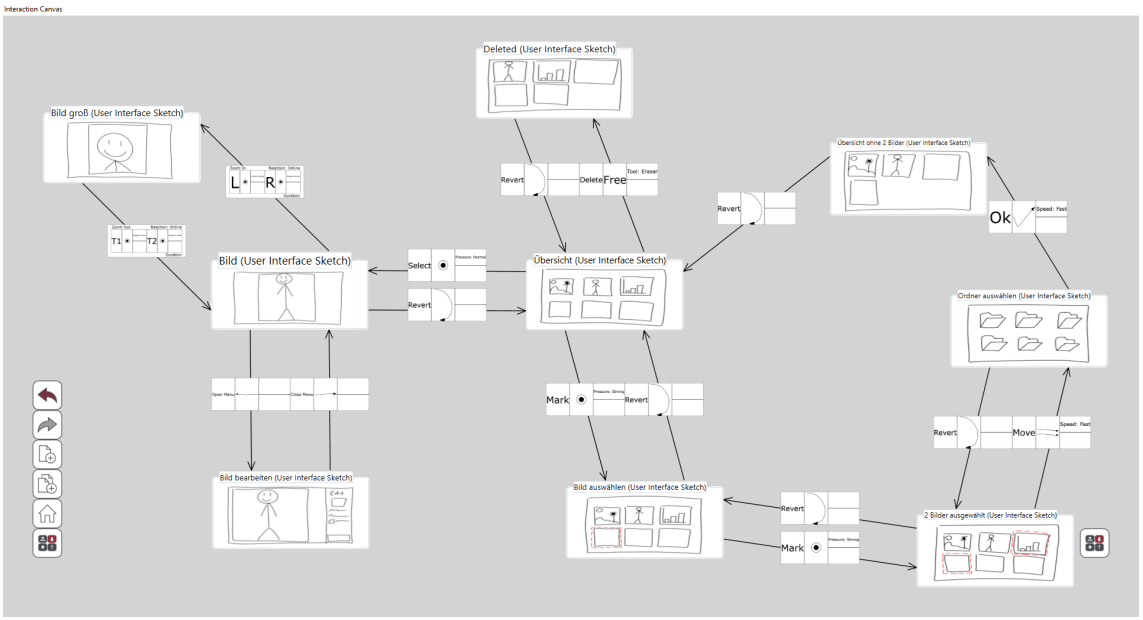

Abbildung 4.7.: Screenshot des digitalen Storyboards im AugIR-Prototyp.

bildmodus geöffnet werden. Die Bearbeitung von UI-Skizzen unterscheidet sich in einigen Punkten von der Bearbeitung generischer Skizzen, weil für eine geöffnete UI-Skizze zusätzlich wie in Abschnitt 3.9.2 erläutert alle verfügbaren Dialogübergänge aufgeführt werden, die von dieser zu anderen Skizzen führen. Dies ist in Abbildung 4.8 beispielhaft durch einen Screenshot illustriert: Von der angezeigten UI-Skizze führen drei Dialogübergänge zu anderen Masken. Diese sind am Bildschirmrand um die Skizze herum angeordnet und jeweils mit der zugehörigen GestureCard versehen. Gekennzeichnete Areale innerhalb der Skizze zeigen an, wo die entsprechenden Gesten auszuführen sind, damit sie den zugeordneten Dialogübergang auslösen.

Über einen Button im Seitenmenü kann der in Abschnitt 3.9.2 beschriebene Simulationsmodus gestartet werden. Hierdurch wird die Bearbeitung der Skizze deaktiviert und die Gestenerkennung aktiviert, so dass alle weiteren Interaktionen mit dem Whiteboard aufgezeichnet und als Gesten interpretiert werden. Wenn eine definierte Geste korrekt erkannt wurde, so wird die zugehörige Transition ausgeführt und die Zielskizze im Vollbildmodus angezeigt. Der Simulationsmodus kann jederzeit beendet werden, um die Bearbeitung der aktuell dargestellten Skizze fortzusetzen.

Abbildung 4.8.: Screenshot der Bearbeitung einer UI-Skizze im AugIR-Prototyp.

Um die Software zu beenden, müssen die Stakeholder zunächst zum Navigationsgraph zurückkehren. Von hier aus können sie die Anwendung auf einem beliebigen Whiteboard schließen, wodurch sie ebenfalls auf allen verbundenen Whiteboards heruntergefahren wird.

4.2. Architektur

Abbildung 4.9 visualisiert die Architektur des AugIR-Prototyps in Form eines UML-Paketdiagramms. Es kommt eine klassische Drei-Schichten-Architektur [35] zum Einsatz:

Die Präsentationsschicht visualisiert die Inhalte des Projekts und stellt über das Paket `GUI` die grafische Benutzerschnittstelle zur Verfügung. Alle geöffneten Applikationsfenster werden dabei von einem Fenstermanager, der durch die Klasse `WindowManager` repräsentiert ist, verwaltet. Benutzereingaben werden in Form von

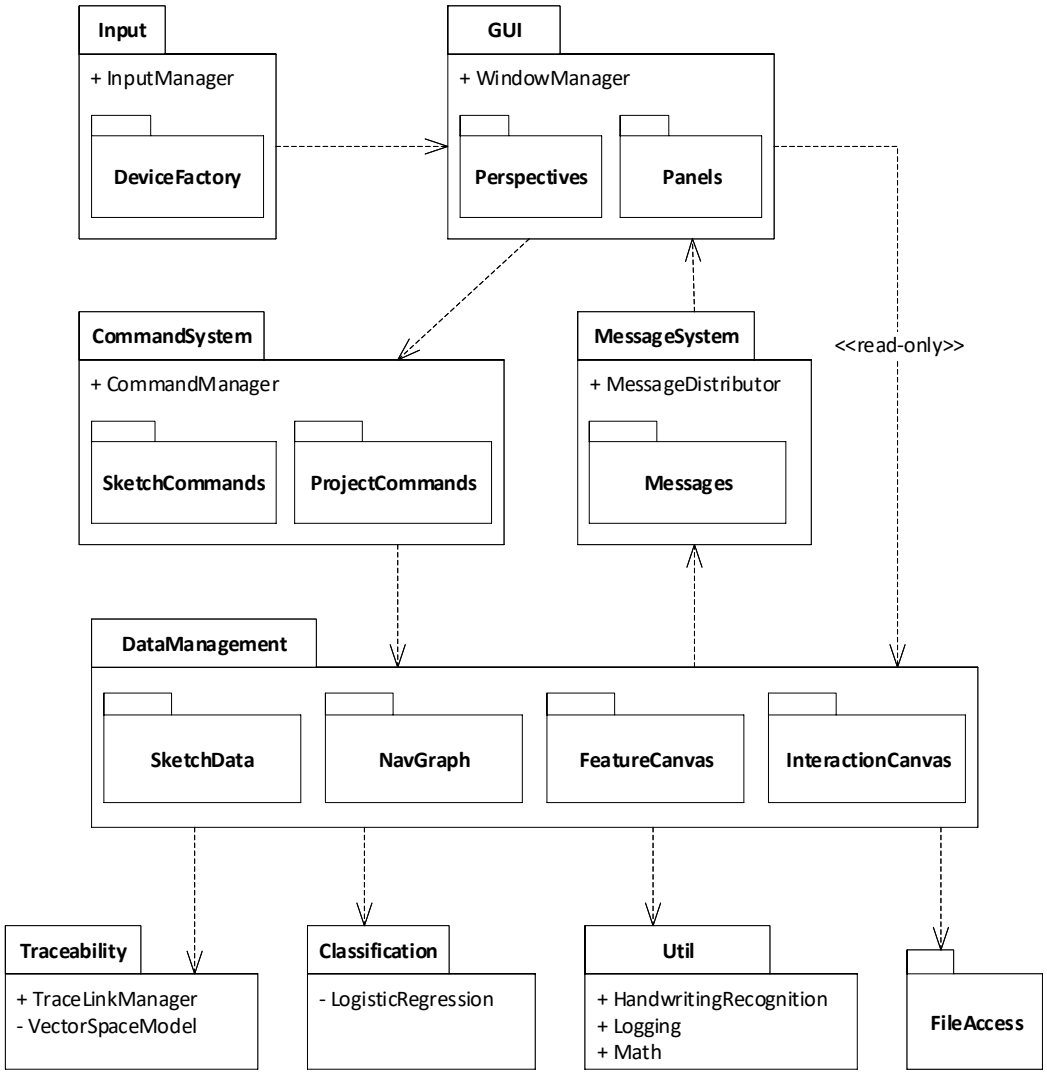

Abbildung 4.9.: UML-Paketdiagramm für den AugIR-Prototyp.

Events vom `InputManager` entgegengenommen und an den Fenstermanager weitergeleitet. Durch Einsatz des Factory-Patterns [78] können je nach angeschlossener Hardware dynamisch unterschiedliche Eingabegeräte angebunden werden. Dies umfasst neben SMART Boards insbesondere auch die Anbindung von Tastatur und Maus, damit die AugIR-Software auch auf Desktop-Computern und Laptops ohne angeschlossene Boards lauffähig ist. Darüber hinaus ermöglicht dies die leichte zukünftige Erweiterung um elektronische Whiteboards anderer Hersteller. Der Aufbau der Präsentationsschicht und der enthaltenen Pakete wird im nachfolgenden Abschnitt 4.2.1 detaillierter beschrieben.

Die Logikschicht enthält die Anwendungslogik des AugIR-Prototyps und besteht im Wesentlichen aus zwei großen Paketen:

Über den `CommandManager` des `CommandSystem`-Pakets kann die UI Befehle absetzen, um auf diese Weise die Datenhaltung zu modifizieren. Hierbei kommt das *Kommando-Entwurfsmuster* (engl. *Command Pattern*) [78] zum Einsatz. Das bedeutet, dass alle Operationen, die von Nutzern der Anwendung ausgeführt werden können, durch sogenannte Kommandos gekapselt sind. Jedes Kommando enthält dabei die zu seiner eigenen Ausführung benötigte Programmlogik. Dies ermöglicht insbesondere die Realisierung eines smarten Undo-/Redo-Systems, weil Kommandos sich auf diese Weise selbst rückgängig machen können. Es wird hierbei zwischen `SketchCommands` und `ProjectCommands` unterschieden. Während erstere sich nur lokal auf eine einzelne Skizze auswirken und deshalb stets ohne Probleme rückgängig gemacht und wiederhergestellt werden können, wirken sich letztere global auf das gesamte Projekt aus und modifizieren in der Regel mehrere Skizzen, weshalb beim Zurücknehmen von Projektkommandos Seiteneffekte zu berücksichtigen sind.

Beide Arten von Kommandos greifen direkt auf die Datenhaltungsschicht zu und können dort die Inhalte von Skizzen sowie den Navigationsgraph, den Feature Canvas und den Interaction Canvas modifizieren. Nach einer Änderung der Daten sendet die Datenhaltung über den `MessageDistributor` des `MessageSystem`-Pakets eine passende Nachricht an die Präsentationsschicht, die detaillierte Informationen über die durchgeführten Modifikationen enthält. Alle UI-Elemente können sich für beliebige Nachrichtentypen registrieren, wenn sie diese empfangen möchten. Nach Emp-

fang einer entsprechenden Nachricht kann die grafische Benutzerschnittstelle darauf reagieren, indem sie beispielsweise die angezeigten Inhalte aktualisiert. Hierzu kann die Präsentationsschicht auch direkt auf die Datenhaltung zugreifen und Daten abfragen, wobei dieser Zugriff jedoch ausschließlich lesend erfolgen darf. Eine Modifikation der Datenhaltung ist der Präsentationsschicht grundsätzlich nur über das Kommandosystem gestattet und wird über entsprechende Interfaces umgesetzt.

Darüber hinaus umfasst der AugIR-Prototyp weitere Pakete zur Verwaltung und Erfassung von Trace Links, zur Klassifizierung von Linienzügen mit Hilfe der logistischen Regression, zum lesenden und schreibenden Zugriff auf Dateien sowie verschiedene Hilfskomponenten zur Handschrifterkennung, Protokollierung von Ereignissen und Durchführung mathematischer Berechnungen.

Da es den Rahmen dieser Arbeit sprengen würde, jedes Paket im Detail zu betrachten, soll im Folgenden exemplarisch das `GUI`-Paket vorgestellt werden, welches die wesentlichen Komponenten der Präsentationsschicht umfasst.

4.2.1. Präsentationsschicht

Abbildung 4.10 illustriert einen Ausschnitt der Präsentationsschicht im AugIR-Prototyp als UML-Klassendiagramm.

Weil die AugIR-Software mehrere miteinander verbundene elektronische Whiteboards unterstützt, werden die Inhalte in mehreren Fenstern visualisiert. Diese werden im Vollbildmodus dargestellt, wobei in der Regel auf jedem Whiteboard ein Fenster angezeigt wird. Es können aber auch zu jeder Zeit beliebig viele Fenster auf dem gleichen Whiteboard geöffnet und nebeneinander angeordnet werden, um beispielsweise unterschiedliche Stände einer Skizze gegenüberzustellen und miteinander zu vergleichen. Alle Fenster im AugIR werden durch die Basisklasse `AugIRWindow` repräsentiert, die von der WPF-Klasse `Window` abgeleitet ist.

Alle geöffneten Fenster werden von einem Fenstermanager verwaltet, der durch die Klasse `WindowManager` realisiert ist. Hierbei handelt es sich um ein Singleton [78], so dass zur Laufzeit der Applikation stets immer nur eine Instanz dieser Klasse

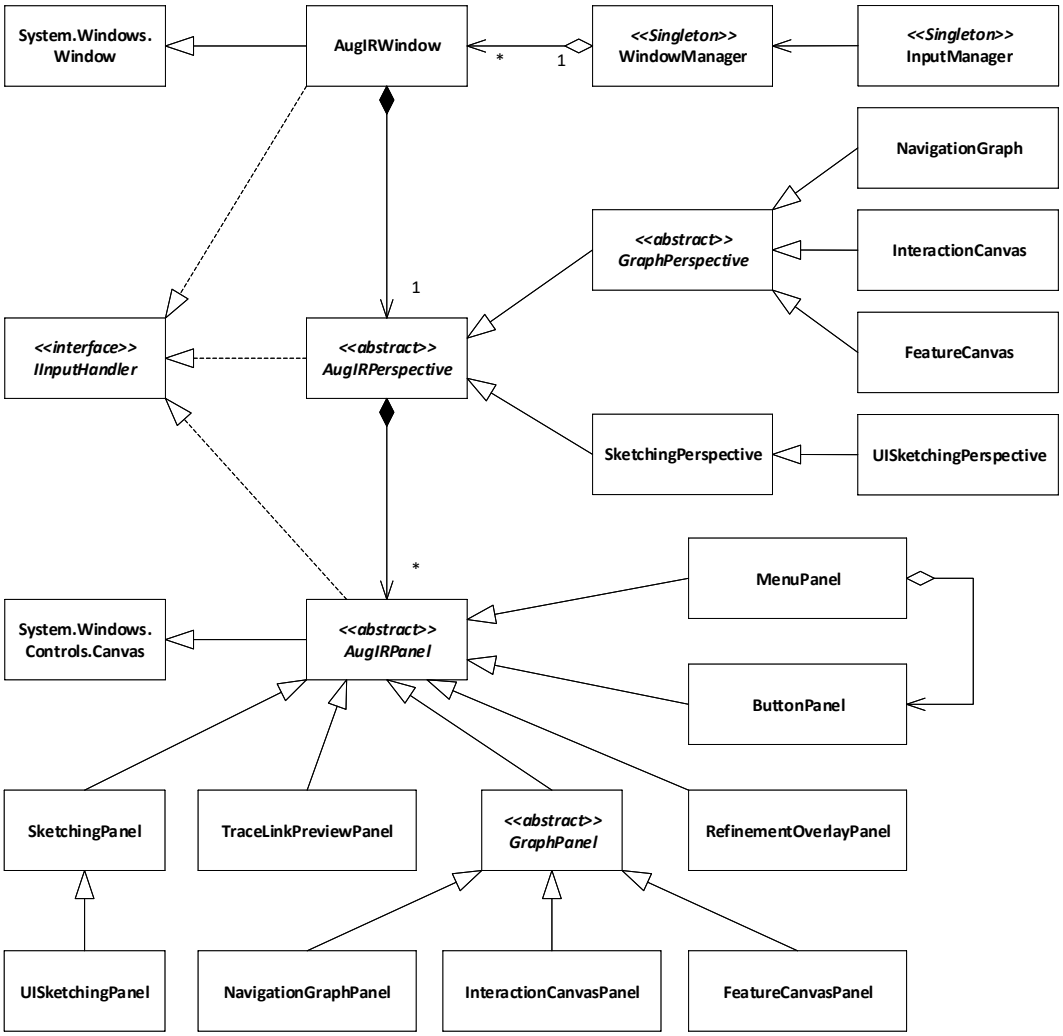

Abbildung 4.10.: UML-Klassendiagramm für Elemente der Präsentationsschicht.

existiert. Diese entscheidet, welches Fenster auf welchem Whiteboard an welcher Position dargestellt wird. Darüber hinaus nimmt sie sämtliche Eingabeevents der Benutzer, die von der Singleton-Instanz der Klasse `InputManager` empfangen und verarbeitet werden, entgegen und leitet sie an das passende Fenster weiter.

Zur Darstellung der eigentlichen Inhalte besteht jedes Fenster im Wesentlichen aus einer sogenannten Perspektive, die durch die abstrakte Klasse `AugIRPerspective` realisiert ist. Eine Perspektive beschreibt den Aufbau eines Fensters und dient als Container für weitere Steuerelemente.

Im AugIR gibt es zwei grundlegende Arten von Perspektiven:

Die abstrakte Klasse `GraphPerspective` ist die Basis für alle Ansichten, die in irgendeiner Form eine Art von Graph repräsentieren. Hierbei handelt es sich in naheliegender Weise zum einen um den Navigationsgraph. Dieser wird durch die Klasse `NavigationGraph` repräsentiert und enthält Skizzen- und Gruppenknoten, die über verschiedene Arten von Verbindungen miteinander verknüpft sein können. Ebenso ist das digitale Storyboard des Interaction Canvas als Graph realisiert und durch die Klasse `InteractionCanvas` umgesetzt. Knoten in diesem Graphen repräsentieren UI-Skizzen, während die Kanten für Dialogübergänge stehen, die durch Gesten ausgelöst werden können. Darüber hinaus ist ebenfalls der Feature Canvas als Graph implementiert und wird durch die Klasse `FeatureCanvas` realisiert. Hierbei handelt es sich um einen nicht-zusammenhängenden Graphen, der keine Kanten besitzt. Die Knoten sind jeweils die einzelnen Feature-Karten, die auf den mobilen Endgeräten erstellt und drahtlos auf die elektronischen Whiteboards übertragen wurden.

Die andere Art von Perspektive dient zur eigentlichen Skizzierung der Inhalte. Sie wird durch die Klasse `SketchingPerspective` implementiert und stellt entsprechende Funktionalitäten zur Erstellung von Skizzen zur Verfügung. Darüber hinaus dient sie als Basisklasse für eine weitere Perspektive, die zur Modellierung von UI-Skizzen verwendet wird. Diese ist durch die Klasse `UISketchingPerspective` realisiert. Sie überschreibt bestimmte Methoden der Basisklasse und stellt insbesondere den Simulationsmodus zur interaktiven Ausführung von UI-Skizzen zur Verfügung.

Die Verwendung einer gemeinsamen Basis für alle Perspektiven hat den wesentlichen Vorteil, dass grundlegende Funktionalitäten nur einmalig in der abstrakten Basisklasse implementiert werden müssen und in allen Subklassen zur Verfügung stehen. Dadurch verfügen beispielsweise alle Perspektiven im AugIR automatisch über eine stufenlose Zoom-Funktion und eine theoretisch unbegrenzte Arbeitsfläche, ohne dass diese Funktionalitäten explizit in den jeweils abgeleiteten Klassen erneut zu implementieren wären.

Jede Perspektive besteht aus einer oder mehreren Gruppen von Steuerelementen, mit denen die Benutzer der Anwendung interagieren können. Diese sogenannten Panels werden durch die abstrakte Klasse `AugIRPanel` modelliert, die von der WPF-Basisklasse `Canvas` abgeleitet ist. Es handelt sich hierbei vereinfacht ausgedrückt um einen Container, der eine beliebige Anzahl von Steuerelementen enthält und verwaltet.

Die AugIR-Software definiert eine Vielzahl unterschiedlicher Panels für verschiedene Anwendungszwecke. Jedes Panel kann hierbei dynamisch zur Laufzeit einer Perspektive hinzugefügt und an einer beliebigen Stelle auf dem Bildschirm platziert werden. Die meisten Perspektiven enthalten mehrere Panels, die nebeneinander oder übereinander dargestellt werden. In der Regel entscheidet eine Perspektive selbst, welche Panels sie an welcher Stelle anzeigen möchte.

Für jede Perspektive existiert ein gleichnamiges Hauptpanel, welches das gesamte Fenster umspannt und den Rahmen bzw. Hintergrund für alle anderen Steuerelemente bildet. Darauf können sich weitere Panels befinden, welche die Durchführung spezieller Aktionen ermöglichen. Die Klasse `TraceLinkPreviewPanel` realisiert beispielsweise das Vorschaufenster für Trace Links, welches sich öffnet, wenn ein unterstrichenes Element doppelt angetippt wird. Die Klasse `RefinementOverlayPanel` implementiert das Overlay zur Anzeige von Verfeinerungen und Vergröberungen.

Darüber hinaus werden zur Steuerung der AugIR-Anwendung am linken und rechten Rand des Whiteboards Menüleisten mit grafischen Symbolen angezeigt. Diese werden durch die Klasse `MenuPanel` realisiert. Jedes Menü besteht dabei aus einer vertikalen Auflistung von Buttons, die jeweils durch die Klasse `ButtonPanel` umgesetzt sind. Buttons können überdies aber auch ohne umgebendes Menü in einer

Perspektive platziert werden. Sie realisieren beispielsweise die in Abschnitt 3.11.2 beschriebene Möglichkeit, vorwärts und rückwärts durch die Historie aller bisher geöffneten Skizzen zu navigieren.

Die drei Klassen `AugIRWindow`, `AugIRPerspective` und `AugIRPanel` implementieren jeweils das Interface `IInputHandler`. Dieses definiert Methoden zur einheitlichen Behandlung von Eingabeevents und erlaubt eine konsistente Verarbeitung und Weiterleitung von Benutzerinteraktionen vom Fenstermanager bis hinunter in die Panels, die konkret auf diese Eingaben reagieren müssen.

4.2.2. Eingabeverarbeitung

Abbildung 4.11 zeigt ein UML-Sequenzdiagramm, das in vereinfachter Form die Erstellung eines Fensters und die Verarbeitung einer Benutzerinteraktion auf einem SMART Board visualisiert.

Nach dem Start des AugIRs werden zunächst ein oder mehrere Fenster erzeugt und geöffnet. Jedes Fenster wird beim `WindowManager` registriert, so dass er dieses verwalten kann. Der Fenstermanager registriert seinerseits alle Fenster beim `InputManager`, so dass dieser später Benutzereingaben entgegennehmen und an den Fenstermanager weiterleiten kann.

Sobald ein Benutzer mit dem elektronischen Whiteboard via Finger, Stift oder Radierer interagiert, empfängt der `InputManager` in der Windows-Nachrichtenschleife ein entsprechendes Event. In der aktuellen Version kann der AugIR neben Eingaben über das SMART Board insbesondere auch auf Eingaben über Tastatur und Maus reagieren, so dass die AugIR-Software grundsätzlich auch auf einem Laptop, Desktop-Computer oder Windows-Tablet ohne angeschlossenes elektronisches Whiteboard lauffähig ist. Für jede mögliche Eingabequelle existiert hierzu eine eigene Klasse, die das entsprechende Eingabegerät verwaltet und Benutzerinteraktionen abfängt.

Sobald eine solche Interaktion vom `InputManager` registriert wird, leitet er das entsprechende Eingabeevent an den `WindowManager` weiter. Zuvor werden die Eingaben

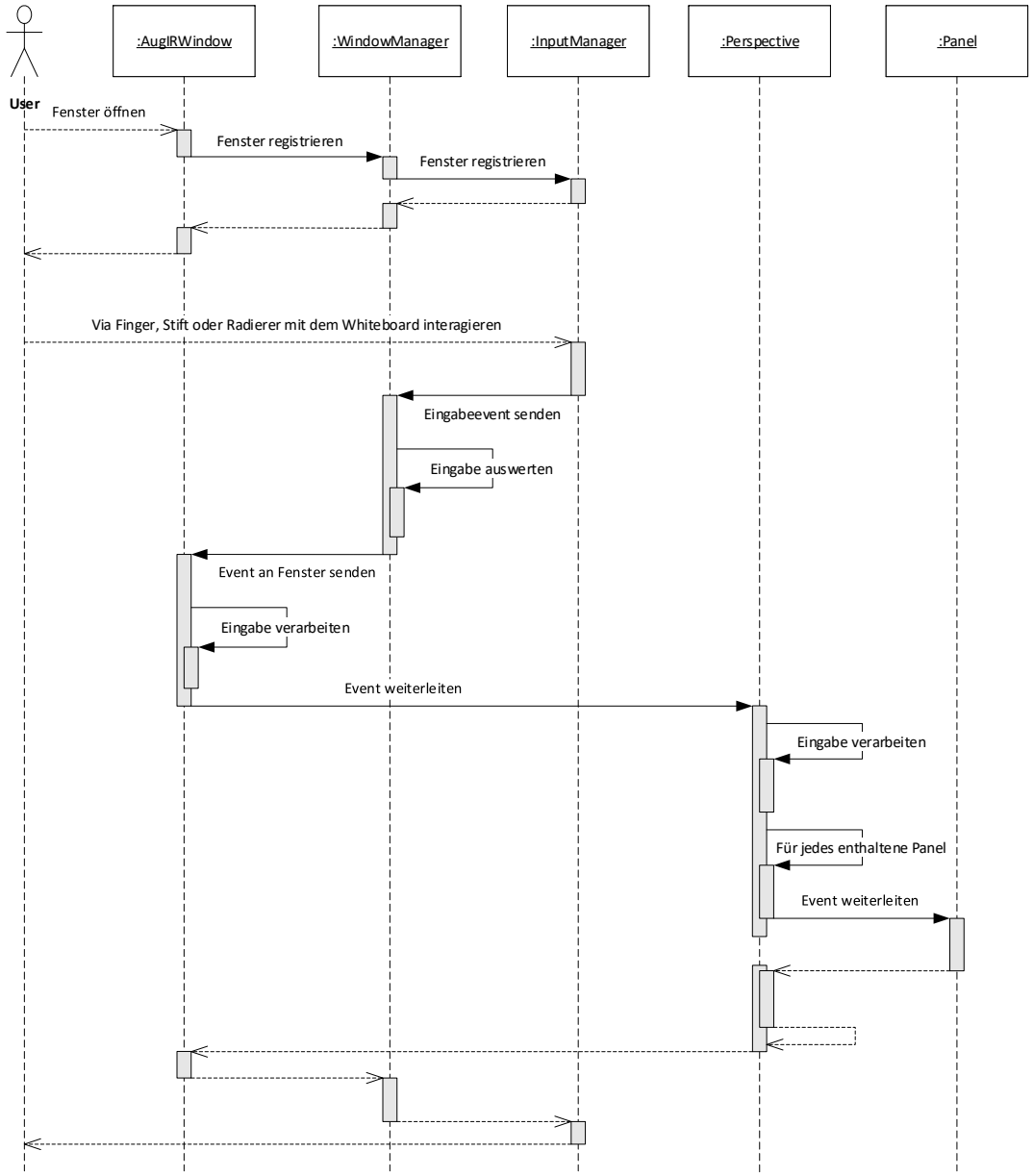

Abbildung 4.11.: UML-Sequenzdiagramm zur Verarbeitung von Benutzerinteraktionen.

in ein einheitliches Datenformat überführt, so dass der Fenstermanager nicht weiß – und insbesondere auch nicht wissen muss – von welcher konkreten Quelle eine Eingabe stammt. Auf diese Weise kann die AugIR-Software auch zukünftig flexibel um jegliche denkbaren Eingabegeräte erweitert werden, ohne dass hierzu Klassen in einer höheren Schicht angepasst werden müssen.

Der Fenstermanager wertet die Eingabe aus und prüft zunächst, ob es sich um eine einzelne Interaktion handelt oder ob diese zusammen mit vorherigen Interaktionen eine Geste bildet. Die entsprechende Eingabeinformation wird dann an das passende Fenster weitergeleitet, in welchem die Interaktion ausgeführt wurde.

Das Fenster kann nun zunächst eigenständig auf die Benutzereingabe reagieren, bevor es diese als Event an die dargestellte Perspektive weiterleitet. Die Perspektive verfährt analog, indem sie ebenfalls zunächst die Eingabe verarbeitet und anschließend der Reihe nach an alle enthaltenen Panels weiterreicht, so dass diese ebenfalls individuell auf die Interaktion reagieren können.

Die vorangegangenen Erläuterungen machten deutlich, dass es sich bei der AugIR-Software weniger um eine einfache Anwendung zur Erstellung digitaler Skizzen sondern vielmehr um ein vollständiges Framework handelt, mit dem Zeichenapplikationen für elektronische Whiteboards entwickelt werden können. Aus diesem Grund ist das gesamte Softwareprojekt bewusst generisch aufgebaut, so dass es sich zukünftig leicht um weitere Perspektiven, Panels und Eingabemodalitäten erweitern lässt.

Teil III.

Evaluation

5. Erstellung und Annotierung von Skizzen

In diesem Kapitel wird das Ergebnis einer ersten Nutzerstudie zum AugIR präsentiert. Der Fokus liegt hierbei auf der Erstellung und Annotierung von Freihandskizzen. Es soll untersucht werden, in wie fern der AugIR Stakeholder bei diesen Tätigkeiten unterstützen kann und ob die Verwendung der elektronischen Whiteboards grundsätzlich intuitiv ist. Dies ist eine wichtige Grundvoraussetzung dafür, dass Stakeholder digitale Zeichenflächen als Ersatz für traditionelle analoge Whiteboards akzeptieren.

Die AugIR-Software wurde hierzu über einen Zeitraum von 12 Monaten bei einem großen deutschen IT-Dienstleiter zur Durchführung von Mitarbeiterschulungen eingesetzt. In sieben jeweils zweitägigen Schulungen wurde verschiedenen Teilnehmern die Interaction-Room-Methodik vermittelt. Dabei mussten sie gemeinsam eine umfangreiche Fallstudie bearbeiten und verschiedene Freihandskizzen auf digitalen Whiteboards erstellen, annotieren und diskutieren. Nach jeder Schulung wurden die Teilnehmer gebeten, einen Fragebogen zur Bewertung ihrer subjektiven Erfahrungen mit dem AugIR auszufüllen.

Im Folgenden wird zunächst die Fallstudie erläutert, die im Rahmen dieser Schulungen zu bearbeiten war. Danach wird die Auswertung der vorgenannten Fragebögen präsentiert. Das Kapitel schließt mit einer kurzen Betrachtung der Gefahren für die Gültigkeit und einer Zusammenfassung der gewonnenen Erkenntnisse.

5.1. Aufbau und Durchführung der Fallstudie

In jeder Schulung mussten die Teilnehmer in Gruppen von neun bis zwölf Personen eine umfangreiche zweitägige Fallstudie bearbeiten, um das vermittelte Wissen praktisch anzuwenden und zu vertiefen. Hierbei sollten sie gemäß der Interaction-Room-Methodik gemeinschaftlich verschiedene Freihandskizzen auf den digitalen Whiteboards im AugIR erstellen, annotieren und diskutieren. In jeder Schulung wurde die gleiche Fallstudie mit identischer Aufgabenstellung bearbeitet, so dass die gewonnenen Ergebnisse vergleichbar sind.

Die Aufgabe bestand jeweils darin, einen fiktiven Anforderungsworkshop für die Deutsche Bahn AG durchzuführen, bei dem die Prozessabläufe zur Bestellung, Bezahlung und Kontrolle von Fahrkarten analysiert und optimiert werden sollten. Die Schulungsteilnehmer mussten hierzu verschiedene Rollen übernehmen: Dies umfasste Stakeholder aus verschiedenen Kundensegmenten, Zugbegleiter, Sachbearbeiter und Anwendungsentwickler. Darüber hinaus schlüpften jeweils zwei Teilnehmer in die Rolle der Interaction-Room-Coaches und moderierten den fiktiven Workshop.

Der Ablauf der Fallstudie orientierte sich am typischen Aufbau von Interaction-Room-Workshops, wie er in Abschnitt 1.2 beschrieben ist: Zunächst wurde ein Feature Canvas erstellt, indem alle Teilnehmer individuell Feature-Karten schrieben und anschließend gemeinschaftlich diskutierten und priorisierten. Die wichtigsten Features wurden danach als Prozesse auf Process Canvases skizziert. Hierbei wurden relevante Fachobjekte gesammelt und auf einem Object Canvas zueinander in Beziehung gesetzt. Danach wurden Schnittstellen und Abhängigkeiten zu umliegenden Systemen auf einem Integration Canvas modelliert. Abschließend wurden die erstellten Freihandskizzen mit Interaction-Room-Annotationen versehen.

An den insgesamt sieben Schulungen nahmen in Summe 55 Teilnehmer teil. Die meisten waren erfahrene Workshop-Moderatoren und Consultants, die bereits umfassende Erfahrungen mit der Arbeit an analogen Whiteboards besaßen. Fast alle verfügten über mindestens rudimentäre Kenntnisse in einer Modellierungssprache wie beispielsweise UML oder BPMN.

Darüber hinaus hatten die Teilnehmer jedoch sehr heterogene Wissensstände und Hintergründe: Die Gruppen bestanden aus professionellen Softwareingenieuren, Requirements Engineers, UI-Designern sowie Beratern aus verschiedenen Domänen wie Automotive, Banking und Versicherungen.

Nach Abschluss der Fallstudie füllte jeder Teilnehmer anonym einen Fragebogen aus. Die entsprechenden Ergebnisse werden im folgenden Abschnitt vorgestellt.

5.2. Ergebnisse

Nachfolgend werden die Ergebnisse der ausgefüllten Fragebögen zur Bewertung der Erstellung und Annotierung von Skizzen im AugIR präsentiert. Jeder Fragebogen enthielt 13 zu bewertende Aussagen sowie ein Freitextfeld, in dem die Teilnehmer beschreiben konnten, was ihnen am AugIR besonders gut oder gar nicht gefiel.

Um die Teilnehmer beim Ausfüllen des Fragebogens nicht durch die Formulierung der Fragestellungen zu beeinflussen, wurden diese bewusst neutral gewählt. Anstatt einer wertenden Aussage zuzustimmen oder diese abzulehnen, konnte eine objektive aber unvollständige Aussage auf einer 5-stufigen Likert-Skala [144] durch eine passende Wertung komplettiert werden.

Hierbei wurde bewusst eine ungerade Anzahl von Antwortmöglichkeiten vorgegeben, damit unentschlossene Teilnehmer auch eine neutrale Antwort wählen konnten. Weil gezeigt werden soll, dass die digitalen Zeichenflächen im AugIR mindestens so gut funktionieren wie analoge Whiteboards, ist es in vielen Fällen ausreichend, wenn beide Werkzeuge als gleich gut bewertet werden.

Das Ausfüllen der Fragebögen geschah vollständig anonym und ohne Aufsicht, um möglichst objektive Antworten der Teilnehmer zu erhalten.

Die Ergebnisse der Befragung sind in Tabelle 5.1 dargestellt.

Tabelle 5.1.: Fragebogen zur Bewertung der Erstellung und Annotierung von Skizzen im AugIR.

#	Aussage	Bewertung				
		++	+	o	-	--
1	Die Bearbeitung von Skizzen im AugIR ist...	Intuitiv				Nicht intuitiv
		13	38	4	0	0
2	Die Bearbeitung von Skizzen im AugIR ist...	Schnell			Zeitlich aufwändig	
		18	31	6	0	0
3	Das Platzieren von Annotationen im AugIR ist...	Intuitiv				Nicht intuitiv
		31	19	5	0	0
4	Das Platzieren von Annotationen im AugIR ist...	Schnell			Zeitlich aufwändig	
		31	19	4	1	0
5	Insgesamt empfinde ich die Bedienung des AugIRs als...	Intuitiv				Nicht intuitiv
		12	38	5	0	0
6	Den Umgang mit dem AugIR zu erlernen war...	Leicht				Schwierig
		24	25	6	0	0
7	Verglichen mit analogen Whiteboards ist die Bearbeitung von Skizzen im AugIR...	Intuitiver			Weniger Intuitiv	
		7	19	21	8	0

Tabelle 5.1.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
8	Verglichen mit analogen Whiteboards ist die Bearbeitung von Skizzen im AugIR...	Schneller				Zeitlich aufwändiger
		28	15	10	2	0
9	Verglichen mit analogen Whiteboards hat der AugIR mich bei der Bearbeitung der Skizzen...	Unterstützt				Behindert
		28	19	7	1	0
10	Verglichen mit analogen Whiteboards ist das Platzieren von Annotationen im AugIR...	Intuitiver				Weniger intuitiv
		4	18	29	3	1
11	Verglichen mit analogen Whiteboards ist das Platzieren von Annotationen im AugIR...	Schneller				Zeitlich aufwändiger
		5	17	26	6	1
12	Verglichen mit analogen Whiteboards hat der AugIR mich beim Platzieren von Annotationen...	Unterstützt				Behindert
		8	26	18	3	0
13	Insgesamt halte ich den AugIR für...	Nützlich				Nicht nützlich
		27	25	3	0	0

Aus der Tabelle geht insgesamt hervor, dass fast alle Teilnehmer die Bedienung des AugIRs als intuitiv empfinden und der Umgang mit der Software für sie leicht zu erlernen war. Dabei betrachten sie die Bearbeitung und Annotierung der Skizzen als intuitiv und schnell. Die meisten Teilnehmer sind der Meinung, dass der AugIR sie

bei diesen Tätigkeiten sinnvoll unterstützt habe. Insgesamt halten deshalb fast alle Teilnehmer den AugIR für nützlich oder sehr nützlich. Kein einziger Teilnehmer sah den AugIR als nicht nützlich an.

Zusätzlich zur Bewertung der vorgegebenen Aussagen konnten die Teilnehmer in einem Freitextfeld beschreiben, was ihnen am AugIR besonders gut oder gar nicht gefiel. Die wesentlichen Antworten, die von mehreren Teilnehmern unabhängig voneinander gegebenen wurden, werden im Folgenden kurz zusammengefasst:

Viele Teilnehmer lobten die einfache Bedien- und Erlernbarkeit der Software. Die Erstellung von Skizzen sei intuitiv und schnell möglich. Dennoch biete der AugIR eine sinnvolle Unterstützung bei der Bearbeitung der digitalen Inhalte, indem gezeichnete Elemente beispielsweise selektiert und frei auf der Zeichenfläche verschoben werden können. Dies ermögliche eine leichte Anpassung existierender Inhalte, ohne dass hierfür gegebenenfalls große Teile einer Skizze neu gezeichnet werden müssen.

Auch die theoretisch unendlich große Zeichenfläche der elektronischen Whiteboards, die durch Gestensteuerung stufenlos gezoomt werden kann, nahmen viele Teilnehmer als positiv wahr. Dadurch, dass die verwendeten Gesten aus anderen populären Anwendungen für Multi-Touch-Oberflächen bekannt sind, seien sie darüber hinaus auch für nicht-technische Stakeholder intuitiv verständlich und ausführbar.

Ebenso wurde die einfache und intuitive Navigation zwischen verschiedenen Skizzen positiv hervorgehoben. Hierdurch ergebe sich im Vergleich zu analogen Whiteboards insbesondere ein großer Platzvorteil, weil ein einzelnes elektronisches Whiteboard zur Darstellung und Bearbeitung aller Skizzen ausreiche, während für jede analoge Skizze in der Regel ein separates Whiteboard benötigt würde.

Die in Abschnitt 3.10 beschriebene Möglichkeit, Skizzen zu logisch zusammengehörigen Gruppen zusammenzufassen, wurde ebenfalls von vielen Teilnehmern als nützlich erachtet. Dies erlaube insbesondere die übersichtliche Organisation und Verwaltung komplexer Projekte mit zahlreichen Skizzen.

Darüber hinaus lobten einige Teilnehmer die Möglichkeit, die skizzierten Inhalte direkt aus dem AugIR heraus als Vektorgrafiken zu exportieren, weil dies besonders nützlich für anschließende Dokumentationszwecke sei.

Als negativ bewerteten hingegen viele Teilnehmer die verwendete Hardware. Weil die eingesetzten SMART Boards nicht über kapazitive Displays verfügen, sondern alle Interaktionen über spezielle Kameras erfasst werden, die in den vier Ecken der elektronischen Whiteboards verbaut sind, sei die Erkennung von Benutzereingaben teilweise unpräzise. Als großer Nachteil wurde insbesondere angeführt, dass jeglicher Kontakt mit der Oberfläche eine Aktion auslöse. Viele Teilnehmer berührten beispielsweise unbeabsichtigt das Board mit dem Ärmel ihres Sakkos, wodurch sie ungewollt Elemente auf der Zeichenfläche malten oder modifizierten.

Weiterhin bemängelten einige Teilnehmer die Genauigkeit der Gestenerkennung. Obwohl die Gesten zur Manipulation der dargestellten Inhalte einfach verständlich und intuitiv ausführbar seien, würden sie vom AugIR nicht immer zuverlässig erkannt. Dies beträfe insbesondere die Pinch-Geste zur Änderung der Zoom-Stufe. Teilnehmer selektierten und/oder verschoben deshalb manchmal versehentlich Inhalte auf der Zeichenfläche, obwohl sie stattdessen eigentlich zoomen wollten.

Außerdem kritisierten manche Teilnehmer die gelegentlich langsame Reaktionszeit der AugIR-Software. Weil es sich um einen Prototyp handelt, ist diese nicht explizit auf Geschwindigkeit optimiert. Deshalb kann es bei der Bearbeitung von umfangreichen Skizzen vereinzelt zu Verzögerungen bei der Auswertung der Eingaben kommen, wenn viele interaktive Inhalte gleichzeitig dargestellt werden. Dies äußert sich beispielsweise darin, dass Linienzüge beim Schreiben gelegentlich erst mit einigen Millisekunden Verzögerung auf der Zeichenfläche erscheinen. Dies sei nach einer kurzen Eingewöhnungszeit zwar nicht mehr störend, falle aber bei erstmaliger Verwendung der digitalen Whiteboards zunächst negativ auf.

5.3. Gefahren für die Gültigkeit

Im Rahmen der vorgestellten Fallstudie hatten prinzipiell alle Teilnehmer die Möglichkeit, ausgiebig mit den elektronischen Whiteboards im AugIR zu arbeiten. Aufgrund der Gruppengröße konnte jedoch nicht jeder exakt im gleichen Umfang davon Gebrauch machen. Einige Teilnehmer interagierten deshalb intensiver und häufiger mit den digitalen Boards als andere. Zwar mussten alle Teilnehmer die Skizzen

individuell annotieren, weshalb diese Funktion von allen gleichermaßen umfassend genutzt wurde, jedoch verbrachten unterschiedliche Teilnehmer unterschiedlich viel Zeit mit der Skizzierung und Bearbeitung der Inhalte. Teilnehmer, die intensiver und häufiger auf den Boards zeichneten als andere, können die Erstellung von Skizzen somit möglicherweise angemessener beurteilen.

Darüber hinaus kann nicht gänzlich ausgeschlossen werden, dass die Bewertungen einiger Teilnehmer besonders positiv ausfielen, weil sie den Trainern hiermit einen Gefallen tun wollten. Um diese Gefahr so weit wie möglich zu minimieren, wurden die Fragebögen wie eingangs erläutert vollständig anonym ohne Aufsicht ausgefüllt.

Aufgrund dessen lässt sich jedoch nicht nachvollziehen, welchen Einfluss das individuelle Vorwissen eines Teilnehmers auf seine Bewertung des AugIRs hatte. Teilnehmer, die bereits Erfahrung im Umgang mit elektronischen Whiteboards oder Multi-Touch-Oberflächen besaßen, empfanden den AugIR möglicherweise als intuitiver, als solche Teilnehmer, die zuvor ausschließlich mit traditionellen analogen Whiteboards gearbeitet haben.

Es ist allerdings bekannt, dass viele Teilnehmer bereits zuvor umfassende praktische Erfahrung in der Moderation von Workshops und der Arbeit mit analogen Whiteboards, Flip-Charts und Metaplanwänden hatten. Aus diesem Grund kann angenommen werden, dass sie adäquat beurteilen können, ob die elektronischen Whiteboards des AugIRs im Vergleich zu analogen Werkzeugen ausreichend intuitiv sind und ob sie sinnvoll bei der Erstellung und Bearbeitung von Skizzen unterstützen können.

Obwohl das Resultat dieser Studie nicht objektiv mess- und bewertbar ist, lässt die subjektive positive Bewertung der Teilnehmer darauf schließen, dass der AugIR für die meisten Benutzer tatsächlich intuitiv zu bedienen ist und von ihnen als nützlich wahrgenommen wird.

5.4. Zusammenfassung und Diskussion

In diesem Kapitel wurde die Auswertung eines Fragebogens zur Bewertung der Erstellung und Annotierung von Skizzen auf elektronischen Whiteboards präsentiert.

Hierbei wurde deutlich, dass fast alle Probanden den AugIR für ein nützliches Werkzeug hielten.

Die Unterstützung bei der Bearbeitung von digitalen Skizzen wurde von fast allen Teilnehmern als positiv bewertet. Obwohl elektronische Whiteboards nicht als wesentlich intuitiver angesehen wurden als analoge Whiteboards, empfanden nur 8 von 55 Teilnehmern diese als weniger intuitiv. 26 Teilnehmer waren sogar der Ansicht, mit elektronischen Whiteboards intuitiver arbeiten zu können. Darüber hinaus waren sich die meisten Teilnehmer einig, dass eine Bearbeitung der Skizzen auf digitalen Whiteboards schneller möglich sei als auf analogen Whiteboards.

Auch hinsichtlich des Platzierens von Annotationen waren die meisten Teilnehmer der Meinung, dass der AugIR sie hierbei unterstützt habe. Verglichen mit analogen Whiteboards wurde das Platzieren von Annotationen im AugIR jedoch nur als geringfügig intuitiver und schneller bewertet.

Insgesamt kann somit festgehalten werden, dass die meisten Teilnehmer den AugIR als nützliches und intuitives Werkzeug betrachteten. Dies ist eine wichtige Grundvoraussetzung dafür, dass digitale Zeichenflächen als Ersatz für traditionelle analoge Whiteboards akzeptiert werden.

6. Verwendung des digitalen Storyboards

Dieses Kapitel präsentiert das Ergebnis einer zweiten Nutzerstudie zum AugIR, welche die Nützlichkeit des in Abschnitt 3.9.2 eingeführten digitalen Interaction Canvas untersucht. Hierbei soll insbesondere analysiert werden, ob das digitale Storyboard im AugIR Stakeholdern beim Verstehen und Definieren von Gesten und Dialogübergängen hilft und welche Vorteile eine digitale Beschreibung von Gesten gegenüber einer traditionellen analogen Beschreibung hat.

Im Folgenden werden zunächst der Aufbau des Experiments und die zugrundeliegende Aufgabenstellung der durchgeführten Fallstudie erläutert. Im Anschluss daran werden Fragebögen ausgewertet, die nach dem Experiment von den Teilnehmern ausgefüllt wurden. Das Kapitel schließt mit einer kurzen Betrachtung der Gefahren für die Gültigkeit sowie mit einer Zusammenfassung und Diskussion der gewonnenen Erkenntnisse.

6.1. Aufbau und Durchführung der Fallstudie

An der folgenden Fallstudie nahmen 14 Personen teil. Jedem Teilnehmer wurden mehrere Aufgaben gestellt, die der Reihe nach in Einzelarbeit zu bearbeiten waren. Die eine Hälfte der Aufgaben sollte auf analogen Whiteboards im Interaction Room, die andere Hälfte auf elektronischen Whiteboards im AugIR gelöst werden.

Zur Bearbeitung der Aufgaben wurden den Teilnehmern zwei Projekte vorgegeben. Weil die in Abschnitt 3.3 vorgestellte Datenbasis jedoch keine Interaction Canvases

umfasst, wurden hierzu zwei passende fiktive Projekte erdacht. Jedes Projekt wurde dabei durch ein Storyboard beschrieben, welches mehrere UI-Skizzen enthielt. Diese waren über Gesture Links miteinander verknüpft (vgl. Abschnitt 3.9.2), um die Dialogübergänge zwischen den Masken anzuzeigen. Jede Verbindung war dabei mit einer vorgegebenen GestureCard versehen, welche die zur Ausführung der Transition erwartete Geste beschrieb.

Beide Projekte waren bewusst allgemeinverständlich gewählt und an gängige Standardsoftware für mobile Endgeräte angelehnt, um den Teilnehmern der Studie das Verständnis der skizzierten Inhalte zu erleichtern.

Das erste Projekt stellte eine mobile Anwendung zur Verwaltung von Fotos dar. Es bestand aus neun Skizzen, die über insgesamt 16 Dialogübergänge miteinander verbunden waren. Über die Benutzerschnittstelle der Anwendung konnten Bilder betrachtet, bearbeitet, in Ordner einsortiert und gelöscht werden.

Das zweite Projekt war eine mobile Anwendung zum Abspielen von MP3-Musikstücken. Es enthielt sieben Skizzen, die über insgesamt 14 Dialogübergänge miteinander verbunden waren. Benutzer konnten in dieser Anwendung Lieder abspielen, vor- und zurückspulen, die Lautstärke der Wiedergabe ändern sowie den Abspielvorgang pausieren und anhalten.

Beide Projekte hatten eine vergleichbare Komplexität: Die vorgegebenen Gesten im MP3-Player waren geringfügig komplizierter, weil dort insbesondere auch zusammengesetzte Multi-Touch-Gesten verwendet wurden. Demgegenüber enthielt der Foto-Editor nur einfache Gesten, dafür aber zwei zusätzliche Skizzen und Dialogübergänge, um die Schwierigkeit der Aufgaben in beiden Projekten auszubalancieren.

Um einem potentiellen Lerneffekt entgegenzuwirken, wurden alle Teilnehmer des Experiments wie in Tabelle 6.1 dargestellt in zwei Gruppen eingeteilt: Teilnehmer der ersten Gruppe hatten zunächst in Runde 1 zwei Aufgaben im Foto-Editor im Interaction Room zu bearbeiten. Danach mussten sie in Runde 2 die beiden gleichen Aufgaben erneut im MP3-Player im AugIR bearbeiten. Für Teilnehmer der zweiten Gruppe war die Reihenfolge genau umgekehrt. Sie mussten zunächst in Runde 1

Tabelle 6.1.: Einteilung der Probanden in zwei Versuchsgruppen.

	Runde 1		Runde 2	
Gruppe 1	Foto-Editor Aufgabe 1 IR	Foto-Editor Aufgabe 2 IR	MP3-Player Aufgabe 1 AugIR	MP3-Player Aufgabe 2 AugIR
Gruppe 2	Foto-Editor Aufgabe 1 AugIR	Foto-Editor Aufgabe 2 AugIR	MP3-Player Aufgabe 1 IR	MP3-Player Aufgabe 2 IR

zwei Aufgaben im Foto-Editor im AugIR lösen und danach die gleichen Aufgaben in Runde 2 im MP3-Player im Interaction Room. Dem Versuchsaufbau liegt somit ein 2-faktorielles Design zugrunde.

Während die Bearbeitung der Aufgaben im Interaction Room auf analogen Whiteboards ohne Zuhilfenahme technischer Werkzeuge erfolgte, wurden die Aufgaben im AugIR auf elektronischen Whiteboards bearbeitet.

Die 14 Teilnehmer im Alter von 21 bis 42 Jahren hatten unterschiedliche Hintergründe und Kenntnisstände. Es nahmen zwei weibliche und zwölf männliche Probanden teil. Elf Teilnehmer waren wissenschaftliche Mitarbeiter der Universität und verfügten über einen Studienabschluss in Informatik oder verwandten Gebieten, während drei Teilnehmer Studenten der Angewandten Informatik waren. Allen Teilnehmern war das grundsätzliche Konzept eines Storyboards bekannt, jedoch hatte niemand explizite Erfahrung in der Modellierung von grafischen Benutzerschnittstellen oder der Definition von Gesten.

Der Ablauf des Experiments war wie folgt:

Um alle Teilnehmer auf den gleichen Wissensstand zu bringen, startete jeder Versuchsdurchlauf zunächst mit einer etwa fünfminütigen Erläuterung von Gesture-Cards. Hierbei erklärte der Moderator des Experiments den in Abschnitt 3.9.2 beschriebenen Aufbau der Karten sowie die zur Verfügung stehenden Eigenschaften.

Darauf folgte eine kurze Erläuterung der Aufgabenstellung. Diese wird im Detail in Abschnitt 6.2 beschrieben.

Anschließend erhielt jeder Teilnehmer eine kurze Einführung in die AugIR-Software. Hierzu erläuterte der Moderator zunächst die wesentlichen Funktionalitäten der Software und demonstrierte deren Verwendung an einem Beispielprojekt. Anschließend durfte der Proband die Software frei ausprobieren und sich mit deren Benutzung vertraut machen.

Danach startete die Bearbeitung der Aufgaben in zwei Runden. Wie eingangs beschrieben bearbeiteten Teilnehmer der ersten Gruppe die Aufgaben in Runde 1 im Interaction Room und in Runde 2 im AugIR, während Teilnehmer der zweiten Gruppe die Aufgaben in Runde 1 im AugIR und danach in Runde 2 im Interaction Room bearbeiteten.

Abschließend füllte jeder Teilnehmer der Studie einen Fragebogen aus. Um die Teilnehmer hierbei nicht zu beeinflussen, geschah das Ausfüllen vollkommen anonym. Alle Fragebögen wurden hierzu ohne Aufsicht ausgefüllt, anschließend zunächst in einer verschlossenen Box gesammelt und erst nach Beendigung des Gesamtexperiments ausgewertet.

6.2. Aufgabenstellung

Wie eingangs erläutert soll im Rahmen dieser Fallstudie überprüft werden, ob der AugIR Stakeholdern beim Verstehen und Definieren von Gesten und Dialogübergängen hilft und ob eine digitale Beschreibung von Gesten Vorteile gegenüber einer traditionellen analogen Beschreibung hat. Hierzu wurde jeder Teilnehmer des Experiments gebeten, in den beiden Projekten jeweils der Reihe nach zwei Aufgaben zu bearbeiten.

Die Reihenfolge der Projekte wurde hierbei nicht alterniert: Beide Gruppen bearbeiteten den Foto-Editor in Runde 1 und den MP3-Player in Runde 2. Die Reihenfolge der Projekte wurde unter Berücksichtigung der Tatsache gewählt, dass die vordefinierten Gesten im Storyboard des Foto-Editors etwas leichter verständlich waren als

beim MP3-Player. Dies ermöglichte den Teilnehmern beider Gruppen einen sanfteren Einstieg in das Experiment.

Die erste Aufgabe bestand darin, einen vorgegebenen Anforderungskatalog mit jeweils sechs User-Stories durchzuarbeiten und zu überprüfen, welche der formulierten Anforderungen im Storyboard des jeweiligen Projekts nicht umgesetzt waren. Hierzu mussten die Teilnehmer sowohl die Inhalte der UI-Skizzen als auch insbesondere die Gesten an den Dialogübergängen gründlich analysieren und verstehen. In beiden Projekten fehlte jeweils die geforderte Möglichkeit, Bilder bzw. Playlisten auf einer Social-Media-Plattform mit Freunden zu teilen. Nach der Identifikation dieses fehlenden Features musste dieses von den Teilnehmern auf dem Storyboard modelliert werden. Hierzu war mindestens eine neue UI-Skizze zu erstellen und über Gesture Links mit den bereits bestehenden Skizzen zu verbinden, um sinnvolle Dialogübergänge zu schaffen.

In der zweiten Aufgabe mussten die Teilnehmer Gesten mit bestimmten Eigenschaften identifizieren und im Storyboard durch passende Gesten mit anderen Eigenschaften ersetzen. Im Foto-Editor waren hierzu alle auf Druck basierenden Gesten auszutauschen, während im MP3-Player alle Multi-Touch-Gesten durch Single-Touch-Gesten zu ersetzen waren. Die Teilnehmer konnten dabei frei entscheiden, wie sie die betreffenden Gesten überarbeiten wollten. Es wurde lediglich gefordert, dass alle Dialogübergänge im Storyboard nach dem Austausch der Gesten immer noch problemlos funktionieren müssen. Hierzu waren die bereits existierenden Gesten zu analysieren und es musste geprüft werden, ob die neuen Gesten zu diesen in Konflikt stehen. Dies ist dann der Fall, wenn von einer Skizze mehrere Transitionen zu anderen Skizzen führen und die Gesten für diese nicht eindeutig unterscheidbar sind, weil dann bei Ausführung der entsprechenden Geste nicht entschieden werden kann, welchem Dialogfluss zu folgen ist.

Weil darüber hinaus den Teilnehmern keine weiteren einschränkenden Vorgaben gemacht wurden, waren die Lösungen beider Aufgaben sehr unterschiedlich und höchst individuell. Aus diesem Grund können sie nicht objektiv gemessen oder bewertet werden. Der Fokus der Fallstudie liegt deshalb darauf, durch die anschließenden Fragebögen zu erfahren, ob das Verstehen und Definieren von Gesten auf elektroni-

schen Whiteboards von den Teilnehmer als leichter und intuitiver wahrgenommen wurde als auf analogen Whiteboards.

6.3. Ergebnisse

Jeder Fragebogen enthielt 22 zu bewertende Aussagen sowie ein Freitextfeld, in dem die Teilnehmer beschreiben konnten, was ihnen am AugIR besonders gut oder gar nicht gefiel.

Genau wie beim Design des Fragebogens für die erste Nutzerstudie, der in Abschnitt 5.2 diskutiert wurde, sind auch die Formulierungen für diesen Fragebogen bewusst neutral gewählt und konnten durch eine Wertung auf einer 5-stufigen Likert-Skala [144] vervollständigt werden.

Erneut geschah das Ausfüllen der Fragebögen vollständig anonym, um möglichst objektive Antworten der Teilnehmer zu erhalten. Deshalb wurden die Fragebögen im Anschluss an jeden Versuchsdurchlauf ohne Aufsicht ausgefüllt und in einer verschlossenen Box gesammelt, die erst nach Beendigung des Gesamtexperiments geöffnet wurde.

Die Ergebnisse der Befragung sind in Tabelle 6.2 dargestellt.

Tabelle 6.2.: Fragebogen zur Bewertung des digitalen Storyboards im AugIR.

#	Aussage	Bewertung				
		++	+	o	-	--
1	Das Verstehen vordefinierter Gesten auf analogen Whiteboards war...	Intuitiv				Nicht intuitiv
		4	7	2	1	0
2	Das Verstehen vordefinierter Gesten auf analogen Whiteboards war...	Schnell				Zeitlich aufwändig
		4	6	2	1	1

Tabelle 6.2.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
3	Das Verstehen vordefinierter Gesten im AugIR war...	Intuitiv				Nicht intuitiv
		8	4	2	0	0
4	Das Verstehen vordefinierter Gesten im AugIR war...	Schnell				Zeitlich aufwändig
		9	3	2	0	0
5	Verglichen mit analogen Whiteboards war das Verstehen vordefinierter Gesten im AugIR...	Intuitiver				Weniger intuitiv
		5	5	4	0	0
6	Verglichen mit analogen Whiteboards war das Verstehen vordefinierter Gesten im AugIR...	Schneller				Zeitlich aufwändiger
		4	4	6	0	0
7	Verglichen mit analogen Whiteboards hat der AugIR mich beim Verstehen vordefinierter Gesten...	Unterstützt				Behindert
		5	5	4	0	0
8	Die Definition neuer Gesten auf analogen Whiteboards war...	Leicht				Schwierig
		6	4	3	1	0
9	Die Definition neuer Gesten auf analogen Whiteboards war...	Schnell				Zeitlich aufwändig
		4	5	2	3	0

Tabelle 6.2.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
10	Die Definition neuer Gesten im AugIR war...	Leicht				Schwierig
		5	4	5	0	0
11	Die Definition neuer Gesten im AugIR war...	Schnell			Zeitlich aufwändig	
		4	6	3	1	0
12	Verglichen mit analogen Whiteboards war die Definition neuer Gesten im AugIR...	Leichter				Schwieriger
		4	4	2	4	0
13	Verglichen mit analogen Whiteboards war die Definition neuer Gesten im AugIR...	Schneller			Zeitlich aufwändiger	
		4	3	2	3	2
14	Verglichen mit analogen Whiteboards hat der AugIR mich bei der Definition neuer Gesten...	Unterstützt				Behindert
		3	7	2	2	0
15	Für das Verstehen der Dialogübergänge war der Simulationsmodus im AugIR...	Hilfreich			Nicht hilfreich	
		10	3	1	0	0
16	Für das Verstehen der Gesten war der Simulationsmodus im AugIR...	Hilfreich			Nicht hilfreich	
		10	2	1	1	0

Tabelle 6.2.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
17	Für das Verstehen der Gesten war die Animation der Gestenbewegung im AugIR...	Hilfreich				Nicht hilfreich
		10	3	1	0	0
18	Insgesamt empfinde ich die Bedienung des AugIRs als...	Intuitiv				Nicht intuitiv
		5	6	3	0	0
19	Den Umgang mit dem AugIR zu erlernen war...	Leicht				Schwierig
		7	7	0	0	0
20	Insgesamt halte ich GestureCards zur Beschreibung von Gesten im AugIR für...	Nützlich				Nicht nützlich
		9	5	0	0	0
21	Insgesamt halte ich den AugIR für...	Nützlich				Nicht nützlich
		10	4	0	0	0
22	Insgesamt würde ich folgendes Werkzeug zur Erstellung von Storyboards präferieren...	AugIR			Analoge Whiteboards	
		4	6	0	3	1

Aus der Tabelle ist ersichtlich, dass die meisten Teilnehmer die vorgegebenen Gesten im AugIR intuitiver und schneller verstehen konnten als auf analogen Whiteboards im Interaction Room. Demgegenüber wurde die Definition neuer Gesten auf elektronischen Whiteboards durchschnittlich jedoch nur als geringfügig leichter und schneller erachtet als die Definition analoger Gesten. Dennoch fühlten sich die meisten Teilnehmer vom AugIR sowohl beim Verstehen als auch beim Definieren von

Gesten unterstützt.

Fast alle Teilnehmer waren der Meinung, dass der Simulationsmodus für das Verstehen von Dialogübergängen und Gesten hilfreich war. Ebenso erachteten sie die Animation der Gestenbewegungen als nützlich.

Die Bedienung des AugIRs wurde dabei von fast allen Teilnehmern als intuitiv oder sehr intuitiv erachtet. Ausnahmslos alle Teilnehmer waren der Meinung, dass für sie der Umgang mit der Software leicht oder sehr leicht zu erlernen gewesen sei.

Insgesamt halten alle Teilnehmer den AugIR für ein nützliches Werkzeug. Die meisten würden ihn zur Erstellung von Storyboards gegenüber analogen Whiteboards bevorzugen.

Zusätzlich zur Bewertung der vorgegebenen Aussagen konnten die Teilnehmer auch in diesem Fragebogen in einem abschließenden Freitextfeld beschreiben, was ihnen am AugIR besonders gut oder gar nicht gefiel. Die wesentlichen Antworten, die von mehreren Teilnehmern unabhängig voneinander gegebenen wurden, werden im Folgenden kurz zusammengefasst:

Sehr viele Teilnehmer lobten ausdrücklich die Animation der Gestenbewegungen auf dem digitalen Storyboard, weil sie diese wesentlich beim Verstehen der Gesten unterstützt habe.

Auch die interaktive Ausführbarkeit der Gesten im Simulationsmodus wurde von vielen Teilnehmern als hilfreich erachtet. Dies vermittele ihnen nicht nur einen besseren Eindruck der Dialogflüsse sondern erlaube es ihnen darüber hinaus ebenfalls, die Funktionalität und Intuitivität der definierten Gesten unmittelbar zu testen.

Einen weiteren großen Vorteil der digitalen Darstellung sahen einige Teilnehmer außerdem in der in Abschnitt 3.9.2 beschriebenen Kennzeichnung der jeweils für eine Geste gültigen Interaktionsfläche innerhalb einer Skizze. Eine entsprechende Visualisierung ist in analogen Storyboards nicht ohne Weiteres möglich, weshalb in diesen nicht immer sofort ersichtlich ist, auf welchen Bereich einer Skizze sich eine Geste bezieht.

Weiterhin wurde auch die einfache und intuitive Bedienung bei der Erstellung und Bearbeitung von UI-Skizzen positiv erwähnt.

Kritik äußerten einige Teilnehmer hingegen insbesondere hinsichtlich der Erstellung neuer Gesten im AugIR. Während eine Modifikation bereits existierender Gesten einfach und schnell möglich sei, müsse zur Definition neuer Gesten ein verhältnismäßig umfangreicher Konfigurationsdialog zur Auswahl und Zusammenstellung passender Gesteneigenschaften verwendet werden. Aus diesem Grund sei die Erstellung neuer Gesten im AugIR oft zeitlich aufwändiger als auf analogen Whiteboards, wo diese einfach auf Papierkarten gezeichnet und mit Magneten an die gewünschte Stelle geheftet werden können.

Auch in dieser Studie kritisierten viele Teilnehmer die verwendete Hardware, weil die Auflösung der verwendeten Kurzdistanz-Beamer verhältnismäßig gering war. Hierdurch seien einige Inhalte sehr schlecht lesbar gewesen, was sich insbesondere bei einer hohen Zoom-Stufe bemerkbar mache. Ein großer Vorteil der digitalen Zeichenflächen sei hierdurch zunichte gemacht worden: Obwohl grundsätzlich alle Inhalte übersichtlich auf einen Blick sichtbar waren, konnten manche Teilnehmer nach dem Herauszoomen des Storyboards einige Inhalte und insbesondere Texte nicht mehr problemlos lesen.

6.4. Gefahren für die Gültigkeit

Es kann nicht gänzlich ausgeschlossen werden, dass die Bewertungen einiger Teilnehmer besonders positiv ausfielen, weil sie den Verantwortlichen hiermit einen Gefallen tun wollten. Um diese Gefahr so weit wie möglich zu minimieren, wurden die Fragebögen wie eingangs erläutert vollständig anonym ohne Aufsicht ausgefüllt.

Weiterhin ist auch ein geschlechtsbezogener Verzerrungseffekt (engl. Gender Bias) nicht auszuschließen, weil nur zwei weibliche aber zwölf männliche Probanden an der Studie teilnahmen und das Ergebnis maßgeblich vom individuellen inhaltlichen Verständnis der Storyboards und der definierten Gesten abhing.

Obwohl die Hintergründe und Kenntnisstände aller Teilnehmer sehr verschieden waren, hatte niemand explizite Vorerfahrung mit der Modellierung von grafischen Benutzerschnittstellen oder der Definition von Gesten, was das Ergebnis ansonsten vermutlich beeinflusst hätte.

Die vorgegebenen Projekte wiesen eine vergleichbare Komplexität auf, waren allerdings thematisch komplett verschieden. Sie waren bewusst an existierende Standardanwendungen für mobile Endgeräte angelehnt. Jedoch ist nicht bekannt, welche Teilnehmer diese Art von Software bereits zuvor selbst nutzten. Mögliche praktische Erfahrung in der Verwendung von Foto-Editoren oder Musikplayern könnte das Verständnis einiger Teilnehmer deshalb gegebenenfalls beeinflusst haben.

Weiterhin hingen die Gesten, welche die Teilnehmer selbst definieren mussten, maßgeblich vom betrachteten Projekt und den bereits vorgegebenen Skizzen und Dialogübergängen ab. Andere Szenarien hätten möglicherweise zu abweichenden Gesten geführt.

Ein großer Kritikpunkt vieler Teilnehmer war die verwendete Hardware und die niedrige Auflösung der Beamer, wodurch einige Inhalte auf dem digitalen Storyboard bei hoher Zoom-Stufe nicht mehr problemlos lesbar waren. Die Teilnehmer wären deshalb möglicherweise bei Verwendung einer besseren Hardware zufriedener gewesen, wodurch einige Antworten gegebenenfalls positiver ausgefallen wären.

6.5. Zusammenfassung und Diskussion

In diesem Kapitel wurde untersucht, ob das digitale Storyboard im AugIR Stakeholdern beim Verstehen und Definieren von Gesten und Dialogübergängen hilft und ob eine digitale Beschreibung von Gesten gegenüber einer traditionellen analogen Beschreibung Vorteile hat. Hierzu wurde die Auswertung eines Fragebogens zur Erfassung der individuellen Einschätzungen von Teilnehmern einer Fallstudie präsentiert.

Es wurde deutlich, dass die meisten Teilnehmer den AugIR grundsätzlich für ein nützliches Werkzeug zur Erstellung von digitalen Storyboards halten.

Gesten waren im AugIR intuitiver und schneller verständlich als auf analogen Whiteboards, was hauptsächlich auf drei Ursachen zurückzuführen ist: Erstens zeigen die digitalen Boards Animationen aller Gestenbewegungen, wodurch diese insbesondere auch für nicht-technische Betrachter leichter nachvollziehbar sind. Zweitens bietet der AugIR einen Simulationsmodus, in dem alle Gesten interaktiv ausprobiert werden können. Dies vermittelt einen besseren Eindruck der verwendeten Gesten und der Dialogflüsse in einer Anwendung als die Betrachtung einer statischen analogen Skizze. Drittens ist im AugIR sofort für jede Geste ersichtlich, auf welchen Bereich einer Skizze sich diese bezieht, weil dieser grafisch hervorgehoben ist. Diese Information fehlt in der Regel in analogen Storyboards, weil sie dort kaum sinnvoll darstellbar ist.

Weiterhin wurde deutlich, dass die Teilnehmer die Definition neuer Gesten im AugIR nur als unwesentlich leichter und schneller erachten als auf analogen Whiteboards. Dies ist absolut plausibel, wenn man bedenkt, dass im analogen Fall einfache Papierkarten beschriftet und mit Magneten an ein Whiteboard geheftet werden können, während im digitalen Fall eine Geste zunächst durch schrittweise Zusammenstellung mehrerer Eigenschaften konfiguriert werden muss. Der Vorteil des AugIRs besteht hierbei jedoch darin, dass die digitalen Menüs eine Hilfestellung bei der Definition von Gesten und der Auswahl passender Eigenschaften bieten, die bei der Verwendung analoger Storyboards nicht pauschal vorhanden ist. Diese Unterstützung wurde von den Teilnehmern als positiv bewertet.

Insgesamt kann somit festgehalten werden, dass das digitale Storyboard im AugIR ausreichend nützlich ist, um von vielen Stakeholdern als Ersatz für traditionelle analoge Whiteboards akzeptiert zu werden. Dennoch ist weitere Forschung notwendig, um die Definition neuer digitaler Gesten noch schneller und intuitiver zu gestalten.

7. Erfassung von Trace Links

In diesem Kapitel wird eine Studie zur Bewertung des in Abschnitt 3.8 vorgestellten Algorithmus zur Erfassung von Trace Links präsentiert. Diese hat einen hohen Stellenwert, weil die Nützlichkeit vieler Lösungsstrategien wie beispielsweise der Unterstützung bei horizontaler Navigation oder der Identifikation von Widersprüchen und Projektrisiken unmittelbar von der Qualität der erfassten inhaltlichen Beziehungen abhängt.

Im Rahmen des nachfolgenden Experiments wird untersucht, wie viele korrekte und falsche Trace Links zwischen Freihandskizzen jeweils für verschiedene Kosinus-Ähnlichkeitsschwellwerte erfasst werden. Als Ergebnis wird ein geeigneter Grenzwert ν bestimmt, mit dem sich hierbei die besten Ergebnisse erzielen lassen. Zudem wird gezeigt, dass die in Abschnitt 3.6 formulierten Tokenisierungsregeln die Qualität der Trace-Link-Erfassung tatsächlich signifikant verbessern. Darüber hinaus wird diskutiert, ob eine vorherige Entfernung von Stoppwörtern aus den erkannten Handschrifttexten sinnvoll ist.

Im Folgenden wird zunächst der Versuchsaufbau erläutert, bevor danach die detaillierten Resultate der Studie präsentiert und diskutiert werden. Das Kapitel schließt mit einer Betrachtung möglicher Gefahren für die Gültigkeit und einer kurzen Zusammenfassung der gewonnenen Erkenntnisse.

7.1. Versuchsaufbau und Durchführung

Die Güte der erfassten Trace Links hängt unmittelbar vom gewählten Kosinus-Ähnlichkeitsschwellwert ν ab. Wird dieser zu hoch gewählt, werden unter Umstän-

den nicht alle relevanten Trace Links erfasst und wichtige Zusammenhänge bleiben unerkannt. Wird er zu niedrig gewählt, können falsche Verknüpfungen zwischen nicht-zusammenhängenden Elementen entstehen.

Um einen optimalen Schwellwert zu bestimmen und dabei die maximal erreichbare Qualität der Trace-Link-Erfassung zu evaluieren, werden im Rahmen eines umfangreichen Experiments Trace Links für verschiedene Schwellwerte für die in Abschnitt 3.3 beschriebenen digital rekonstruierten Interaction-Room-Projekte $P_1, \ldots P_{16}$ erzeugt und ausgewertet. Hierbei wird analysiert, welcher Schwellwert insgesamt die besten Ergebnisse liefert.

Zur Erfassung der Trace Links werden die erkannten Freihandtexte wie in Abschnitt 3.6 beschrieben durch Anwendung der heuristischen Regeln t_1, t_2 und t_3 tokenisiert. Die erzeugten Tokenisierungen werden wie in Abschnitt 3.7 erläutert modifiziert, indem Elemente falls nötig durch hinreichend ähnliche Wörter ersetzt werden. Hierbei wird der in Abschnitt 3.7.2 ermittelte Ähnlichkeitsschwellwert $\mu := 0{,}75$ verwendet.

Die erfassten Trace Links wurden im Rahmen dieses Experiments mit dem folgenden dreistufigen Vorgehen ausgewertet:

1. Im ersten Schritt wurde für jedes Projekt $P_1, \ldots, P_{16}$ eine Referenzliste erstellt, die alle für das jeweilige Projekt korrekten Trace Links enthält. Hierzu wurden alle rekonstruierten Projekte jeweils von drei Softwareingenieuren, die mit der Interaction-Room-Methode vertraut sind, gründlich analysiert. Jeder musste ausführlich aufschreiben und erläutern, zwischen welchen Elementen eines Projekts seiner Meinung nach inhaltliche Zusammenhänge bestehen. Anschließend haben alle drei zusammen ihre individuellen Schätzungen abgeglichen, diskutiert und daraus eine gemeinsame Referenzliste für jedes Projekt erstellt.

2. Im zweiten Schritt wurde der in Abschnitt 3.8 präsentierte Traceability-Algorithmus auf alle Projekte angewendet. Für jedes Projekt wurden dabei der Reihe nach verschiedene Schwellwerte ν aus dem Intervall 0,1 bis 1 verwendet.

3. Im dritten und letzten Schritt wurden die zuvor erfassten Trace Links mit den geschätzten korrekten Trace Links der Referenzlisten aus Schritt 1 verglichen.

Wurde für ein betrachtetes Projekt ein Trace Link erkannt, der nicht in der zugehörigen Referenzliste aufgeführt war, so handelt es sich hierbei um einen falsch erkannten Trace Link. Enthält die Referenzliste hingegen einen Trace Link, der vom Algorithmus nicht erfasst wurde, so handelt es sich um einen fälschlicherweise übersehenen Trace Link.

7.2. Messdatenerhebung

Um die Qualität der erfassten Trace Links messen und bewerten zu können, werden die beiden populären und allgemein akzeptierten Information-Retrieval-Metriken Precision und Recall verwendet [10, 13, 76, 160].

Precision ist ein Messwert für die Genauigkeit der Trace-Link-Erfassung. Sie berechnet sich aus dem Verhältnis der Anzahl der korrekt erfassten Trace Links zur Gesamtzahl aller erfassten Trace Links. Ein Trace Link heißt *falsch positiver Trace Link* oder kurz *False Positive*, falls er vom Traceability-Algorithmus erfasst wurde, obwohl er nicht in der Referenzliste der korrekten Trace Links aufgeführt ist. Eine Precision von 1 zeigt an, dass alle erfassten Trace Links korrekt sind und es keine False Positives gibt. Es können jedoch Trace Links übersehen worden sein, die zwar in einer Referenzliste enthalten sind, vom Algorithmus aber nicht entdeckt wurden. Solche Trace Links, die der Algorithmus fälschlicherweise übersehen hat, werden im Folgenden als *übersehene Trace Links* bezeichnet.

Recall ist ein Messwert für die Trefferquote der Trace-Link-Erfassung. Er beschreibt die Fähigkeit eines Traceability-Algorithmus, so viele korrekte Zusammenhänge wie möglich zu erkennen. Dieser Messwert berechnet sich aus dem Verhältnis der Anzahl der korrekt erfassten Trace Links zur Gesamtzahl der vorab geschätzten Trace Links aus der entsprechenden Referenzliste. Ein Recall-Wert von 1 zeigt an, dass alle geschätzten Trace Links erfasst wurden und es keine übersehenen Trace Links gibt. Es können jedoch falsch positive Trace Links existieren.

Zusammenfassend lauten die formalen Definitionen für Precision und Recall somit wie folgt:

$$\text{Precision} := \frac{\text{Anzahl korrekt erfasster Trace Links}}{\text{Gesamtzahl aller erfassten Trace Links}}$$

und

$$\text{Recall} := \frac{\text{Anzahl korrekt erfasster Trace Links}}{\text{Gesamtzahl der geschätzten Trace Links}}.$$

Grundsätzlich stehen Precision und Recall im Kontrast zueinander. Je höher die Precision ist, desto niedriger ist in der Regel der Recall und umgekehrt. Dies überlegt man sich wie folgt:

Beide Werte hängen unmittelbar mit dem gewählten Schwellwert ν für die Kosinus-Ähnlichkeit zusammen. Wird dieser sehr hoch gewählt, so erfasst der entsprechende Traceability-Algorithmus nur sehr starke und offensichtliche Zusammenhänge. Dies sorgt dafür, dass die Precision sehr hoch ist, weil keine oder nur wenige False Positives erkannt werden. Auf der anderen Seite können hierbei jedoch korrekte Trace Links übersehen werden, deren Kosinus-Ähnlichkeit geringer ist und damit unter dem Schwellwert liegt. Dies kann zu einem niedrigeren Recall führen.

Wird demgegenüber ein niedrigerer Schwellwert ν gewählt, so werden üblicherweise mehr Trace Links erfasst, weil dies auch Zusammenhänge mit geringerer Kosinus-Ähnlichkeit einschließt. Hierdurch kann der Recall steigen, weil damit typischerweise auch mehr korrekte Trace Links erkannt werden. Die Precision kann dabei jedoch signifikant abfallen, weil auf diese Weise häufig auch False Positives (die für gewöhnlich eine niedrigere Kosinus-Ähnlichkeit aufweisen als korrekte Trace Links) erfasst werden.

Es stellt sich somit die Frage, welcher der beiden Werte mit höherer Priorität maximiert werden sollte. Bevor im Folgenden konkrete Messwerte für Precision und Recall untersucht werden, muss deshalb diskutiert werden, welcher Wert für die Trace-Link-Erfassung im AugIR bedeutsamer ist.

Falsch positive Trace Links können fast immer leicht von den beteiligten Stakeholdern identifiziert werden. Häufig handelt es sich dabei um textuell ähnliche Wörter, die fälschlicherweise miteinander verknüpft wurden. Wenn der Traceability-

Algorithmus beispielsweise „Meldung“ und „Anmeldung“ aufgrund ihrer Wortähnlichkeit miteinander verbindet, dann ist eine solche falsche Verknüpfung für die beteiligten Stakeholder häufig sofort erkennbar, ohne dass dafür tiefer gehendes Domänenwissen notwendig wäre. Darüber hinaus besitzt für jeden diskutierten Aspekt eines Projekts häufig mindestens ein anwesender Stakeholder ausreichendes Projektverständnis, um auch nicht-offensichtliche falsch positive Trace Links schnell aufdecken zu können.

Über einen entsprechenden Button in der Vorschauansicht (vgl. Abschnitt 3.11.2) können Stakeholder falsch erkannte Trace Links leicht entfernen. Beziehungen, die auf diese Weise eliminiert wurden, werden einer Blacklist hinzugefügt, so dass die betroffenen Elemente auch in anderen Skizzen zukünftig nicht mehr fälschlicherweise miteinander verbunden werden.

Einer der Hauptgründe für die Anwendung von Traceability ist eine Unterstützung der Stakeholder durch die Identifikation von Zusammenhängen, die leicht übersehen werden können. Folglich ist es wichtig, dass der Algorithmus so viele inhaltliche Beziehungen zwischen den Skizzen wie möglich erfasst.

Es lässt sich argumentieren, dass es deutlich einfacher ist, falsch positive Trace Links einmalig über die Vorschauansicht zu entfernen, als manuell Zusammenhänge aufzudecken, die vom AugIR übersehen wurden. Deshalb liegt der Fokus bei der Trace-Link-Erfassung im Folgenden primär auf einer Maximierung des Recall-Werts, wobei natürlich gleichzeitig eine möglichst hohe Precision angestrebt wird.

Weil sich Precision und Recall wie zuvor erläutert gegenseitig beeinflussen, wird im Kontext des Information Retrievals häufig zusätzlich ein kombiniertes Maß angegeben, dass die Beurteilung der Güte mit einer einzigen Kennzahl erlaubt [13]. Hierzu kann das sogenannte *F-Maß* verwendet werden, welches Precision und Recall durch das gewichtete harmonische Mittel kombiniert und sich wie folgt berechnet [44]:

$$F := 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Weil bei diesem Maß Precision und Recall gleich gewichtet sind, wird es gelegentlich auch als F_1-Maß bezeichnet.

7.3. Ergebnisse

In diesem Abschnitt werden die Ergebnisse des Experiments zur Bewertung der Qualität der Trace-Link-Erfassung beschrieben.

Abschnitt 7.3.1 präsentiert zunächst die maximal erreichbare Genauigkeit und Trefferquote für alle 16 analysierten Interaction-Room-Projekte. Hierbei kommen die in Abschnitt 3.6 formulierten Tokenisierungsregeln t_1, t_2 und t_3 zum Einsatz. Als Ergebnis wird insbesondere ein konkreter Schwellwert ν bestimmt, mit dem sich bei der Trace-Link-Erfassung durchschnittlich die besten Resultate erzielen lassen.

Abschnitt 7.3.2 belegt die Nützlichkeit der Tokenisierungsregeln und zeigt, dass durch deren Anwendung Precision und Recall tatsächlich signifikant verbessert werden können.

Abschließend wird in Abschnitt 7.3.3 erläutert, warum bei der Trace-Link-Erfassung im AugIR keine Entfernung von Stoppwörtern erfolgt.

7.3.1. Genauigkeit und Trefferquote

Abbildung 7.1 gibt zunächst einen Überblick über die durchschnittlichen Werte für Precision, Recall und F-Maß bei Verwendung verschiedener Schwellwerte für die Kosinus-Ähnlichkeit im Bereich 1 bis 0,1 für alle 16 Projekte.

Man sieht, dass sowohl Precision als auch Recall im Bereich 1 bis 0,75 relativ konstant bleiben. Erwartungsgemäß startet die Precision mit einem sehr hohen Wert von 96 %, während der initiale Recall deutlich niedriger bei 62 % liegt. Der Wert des F-Maßes liegt dazwischen und beträgt etwa 71 %.

Bei Absenkung des Kosinus-Ähnlichkeitsschwellwerts auf 0,7 erhöht sich der durchschnittliche Recall signifikant um 14 Prozentpunkte auf 76 %. Gleichzeitig sinkt die Precision leicht um 3 Prozentpunkte auf 93 %. Der Wert von F steigt dabei um 11 Prozentpunkte auf 82 % an.

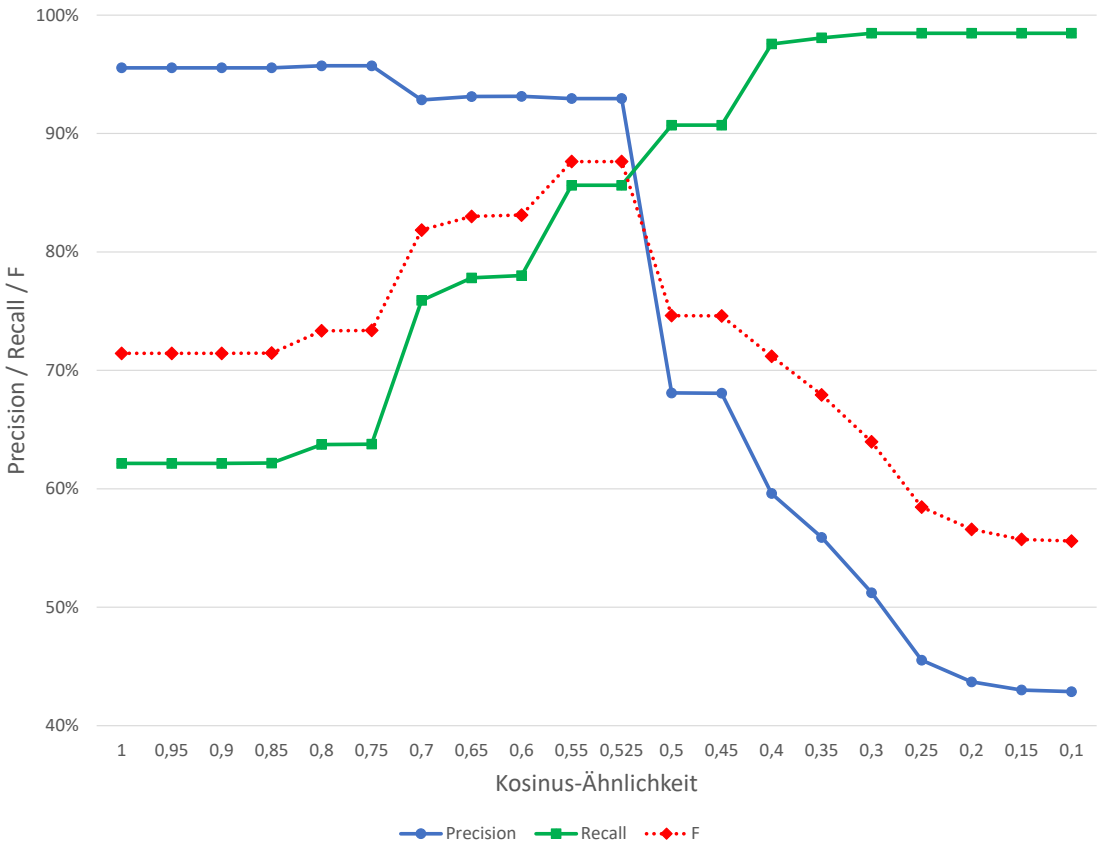

Abbildung 7.1.: Durchschnittliche Werte für Precision, Recall und F-Maß für alle 16 analysierten Projekte bei Verwendung verschiedener Kosinus-Ähnlichkeitsschwellwerte im Bereich 1 bis 0,1.

Im Bereich 0,7 bis 0,525 steigen sowohl Recall als auch F kontinuierlich weiter. Während der Recall in diesem Intervall von 76 % auf 86 % ansteigt, erhöht sich F von 82 % auf 88 %. Die Precision bleibt indessen beinahe konstant und beträgt auch für 0,525 noch etwa 93 %.

Bei einer weiteren Absenkung des Kosinus-Ähnlichkeitsschwellwerts auf 0,5 steigt der Recall zwar um 5 Prozentpunkte auf 91 % an, allerdings fallen Precision und F schlagartig signifikant ab. Die Precision beträgt bei Verwendung dieses Grenzwerts nur noch 68 % und der Wert des F-Maßes fällt auf 75 %.

Erwartungsgemäß sinken Precision und F weiter, je kleiner der Schwellwert gewählt wird. Obwohl die Trefferquote gleichzeitig ansteigt, ist ein derartiges signifikantes Abfallen der Genauigkeit nicht zu rechtfertigen. Eine Absenkung des Schwellwerts unter 0,525 erscheint deshalb nicht sinnvoll.

Um diese Annahme zu verifizieren werden im Folgenden die genauen durchschnittlichen Werte für Precision, Recall und F-Maß für unterschiedliche Kosinus-Ähnlichkeitsschwellwerte detaillierter untersucht. Weil hierbei die im Vorfeld geschätzten Trace Links der Referenzlisten natürlich unabhängig vom gewählten Schwellwert für jedes Projekt konstant bleiben, ist deren Anzahl vorab einmalig in Tabelle 7.1 aufgelistet. Spalte 1 zeigt das Projektkürzel und Spalte 2 die Anzahl der jeweils zugehörigen geschätzten Trace Links. Die letzten beiden Zeilen stellen die Gesamtzahl aller geschätzten Trace Links sowie die durchschnittliche Zahl pro Projekt dar.

Tabelle 7.2 zeigt die durchschnittlichen Werte für Precision, Recall und F für alle 16 analysierten Projekte, die sich bei Verwendung unterschiedlicher Grenzwerte ν zwischen 0,9 und 0,5 ergeben. In der ersten Spalte ist der jeweils verwendete Grenzwert aufgeführt. Daneben steht in der zweiten Spalte die durchschnittliche Anzahl aller erfassten Trace Links für diesen Grenzwert. Dies umfasst sowohl korrekte Trace Links als auch False Positives. Die dritte Spalte stellt die durchschnittliche Anzahl der korrekt erfassten Trace Links dar. Sie gibt also an, wie viele der erfassten Trace Links im Durchschnitt tatsächlich in den vorab erstellten Referenzlisten enthalten sind. In Spalte 4 ist die durchschnittliche Anzahl der Verknüpfungen dargestellt, die vom Algorithmus erfasst wurden, obwohl sie nicht in den Referenzlisten aufgeführt

Tabelle 7.1.: Anzahl der im Vorfeld geschätzten Trace Links für alle 16 Projekte.

Projekt	Geschätzte Trace Links
P_1	42
P_2	38
P_3	10
P_4	18
P_5	50
P_6	19
P_7	70
P_8	45
P_9	17
P_{10}	42
P_{11}	10
P_{12}	32
P_{13}	31
P_{14}	89
P_{15}	17
P_{16}	193
Gesamt	723
Ø	45,19

Tabelle 7.2.: Durchschnittliche Werte für Precision, Recall und F-Maß für verschiedene Grenzwerte ν.

ν	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
0,9	30	28,75	1,25	16,44	0,96	0,62	0,71
0,8	30,69	29,44	1,25	15,75	0,96	0,64	0,73
0,7	38,75	36,38	2,38	8,81	0,93	0,76	0,82
0,6	39,81	37,38	2,44	7,81	0,93	0,78	0,83
0,55	43,88	40,69	3,19	4,5	0,93	0,86	0,88
0,525	44,06	40,69	3,38	4,5	0,93	0,86	0,88
0,5	65,63	42,31	23,31	2,88	0,68	0,91	0,75

sind. Spalte 5 zeigt an, wie viele Trace Links aus den Referenzlisten im Durchschnitt vom Algorithmus nicht gefunden wurden. Die Spalten 6 bis 8 stellen jeweils die durchschnittlichen Werte für Precision, Recall und F für den entsprechenden Grenzwert dar.

Wie bereits eingangs vermutet ist die durchschnittliche Genauigkeit für einen Grenzwert von $\nu = 0{,}9$ mit 0,96 sehr hoch, während die durchschnittliche Trefferquote mit lediglich 0,62 entsprechend niedrig ist. F liegt mit 0,71 dazwischen.

Aus Abbildung 7.1 war bereits ersichtlich, dass diese Werte im Bereich 0,9 bis 0,75 relativ konstant bleiben. Durch Absenkung des Grenzwerts ν auf 0,7 kann der durchschnittliche Recall wesentlich auf 0,76 erhöht werden. Die Precision fällt hierbei im Durchschnitt nur geringfügig auf 0,93 ab, während F für diesen Grenzwert durchschnittlich 0,82 beträgt.

Für $\nu = 0{,}6$ bleibt die Precision unverändert hoch, während der durchschnittliche Recall um zwei Prozentpunkte auf 0,78 ansteigt. Dementsprechend steigt auch der Wert des F-Maßes von 0,82 auf 0,83.

Ein erneutes Absenken des Grenzwerts auf $\nu = 0{,}55$ kann die erzielte Trefferquote bei gleichbleibender Precision signifikant um weitere 8 Prozentpunkt von 0,78 auf 0,86 steigern. Als Folge erhöht sich auch F auf 0,88.

Eine weitere Verringerung des Grenzwerts bringt jedoch keine zusätzliche Verbesserung mehr. Für $\nu = 0{,}525$ scheinen die Werte zunächst zu stagnieren. Allerdings ist aus der Tabelle ersichtlich, dass die durchschnittliche Anzahl der False Positives leicht ansteigt, während die Zahl der korrekt erfassten Trace Links gleich bleibt. Es werden folglich für einige Projekte zusätzliche Trace Links erfasst, die jedoch alle falsch sind.

Abbildung 7.1 ließ bereits vermuten, dass eine Absenkung des Grenzwerts unter 0,525 nicht sinnvoll ist. Die letzte Zeile in Tabelle 7.2 bestätigt dies: Während der Recall zwar auf einen sehr guten Wert von 0,91 ansteigt, fällt die Precision drastisch um 25 Prozentpunkte auf 0,68 ab. Die Wahl von 0,5 als Schwellwert lässt sich somit keinesfalls rechtfertigen, da die Anzahl der falsch positiven Trace Links unakzeptabel hoch ist.

Man kann annehmen, dass ein weiteres Absenken des Kosinus-Ähnlichkeitsschwellwerts zu einer weiteren Verringerung der Precision führt, weil ein niedrigerer Schwellwert in der Regel mit einer steigenden Anzahl von False Positives einhergeht. Dies wird ebenfalls durch Abbildung 7.1 deutlich. Somit besteht keine Notwendigkeit, zusätzliche Kosinus-Ähnlichkeitsschwellwerte kleiner als 0,5 im Detail zu betrachten, weil eine weitere Optimierung der Werte ausgeschlossen ist.

Eine detaillierte individuelle Einzelbetrachtung der Messwerte für jedes Projekt findet sich im Anhang in Abschnitt A.2.

Zusammenfassend konnte somit gezeigt werden, dass bei der Wahl eines Kosinus-Ähnlichkeitsschwellwerts von 0,55 die besten durchschnittlichen Werte für Precision und Recall erreicht werden können. Entsprechend ist für diesen Grenzwert auch der Wert des F-Maßes am höchsten. Es wird deshalb $\nu := 0{,}55$ als Schwellwert für die Kosinus-Ähnlichkeit gewählt. Hierdurch lassen sich eine Precision von 93 % und ein Recall von 86 % erreichen.

7.3.2. Bewertung der heuristischen Tokenisierungsregeln

Bei der zuvor beschriebenen Erfassung von Trace Links kamen die in Abschnitt 3.6 formulierten Tokenisierungsregeln t_1, t_2 und t_3 zur Anwendung. Zur Erinnerung:

- t_1 wandelt alle Buchstaben eines Textes in Kleinbuchstaben um und trennt danach den Text an den Leerzeichen auf.
- t_2 wandelt alle Buchstaben eines Textes in Kleinbuchstaben um und entfernt danach sämtliche Leer- und Sonderzeichen.
- t_3 wandelt alle Buchstaben eines Textes in Kleinbuchstaben um, ersetzt alle Sonderzeichen durch Leerzeichen und trennt den Text abschließend an den Leerzeichen auf.

Im Folgenden wird gezeigt, dass die kombinierte Anwendung aller drei Regeln die Qualität der Trace-Link-Erfassung tatsächlich signifikant verbessern kann.

Hierzu zeigt Abbildung 7.2 zunächst den durchschnittlichen Wert des F-Maßes für alle 16 analysierten Projekte bei Verwendung unterschiedlicher Regelkombinationen.

Man sieht sofort, dass F bei Anwendung aller drei Regeln t_1, t_2 und t_3 für jeden Kosinus-Ähnlichkeitsschwellwert zwischen 1 und 0,1 mit Abstand am größten ist. Demgegenüber ist F bei alleiniger Anwendung von Regel t_1 für $\nu \geq 0{,}45$ durchschnittlich am kleinsten. Bei Verwendung der Regeln t_1 und t_2 liegt F dazwischen bzw. für $\nu < 0{,}45$ leicht darunter.

Erwartungsgemäß ist der durchschnittliche Wert des F-Maßes für den Kosinus-Ähnlichkeitsschwellwert 0,55 am höchsten. Dies deckt sich mit den Erkenntnissen aus Abschnitt 7.3.1

Weil F ein kombiniertes Maß für Precision und Recall darstellt, kann somit bereits anhand von Abbildung 7.2 angenommen werden, dass die Verwendung aller drei Tokenisierungsregeln durchschnittlich die besten Ergebnisse liefert. Dies wird durch Tabelle 7.3 bestätigt, die konkrete Werte für Precision, Recall und F für

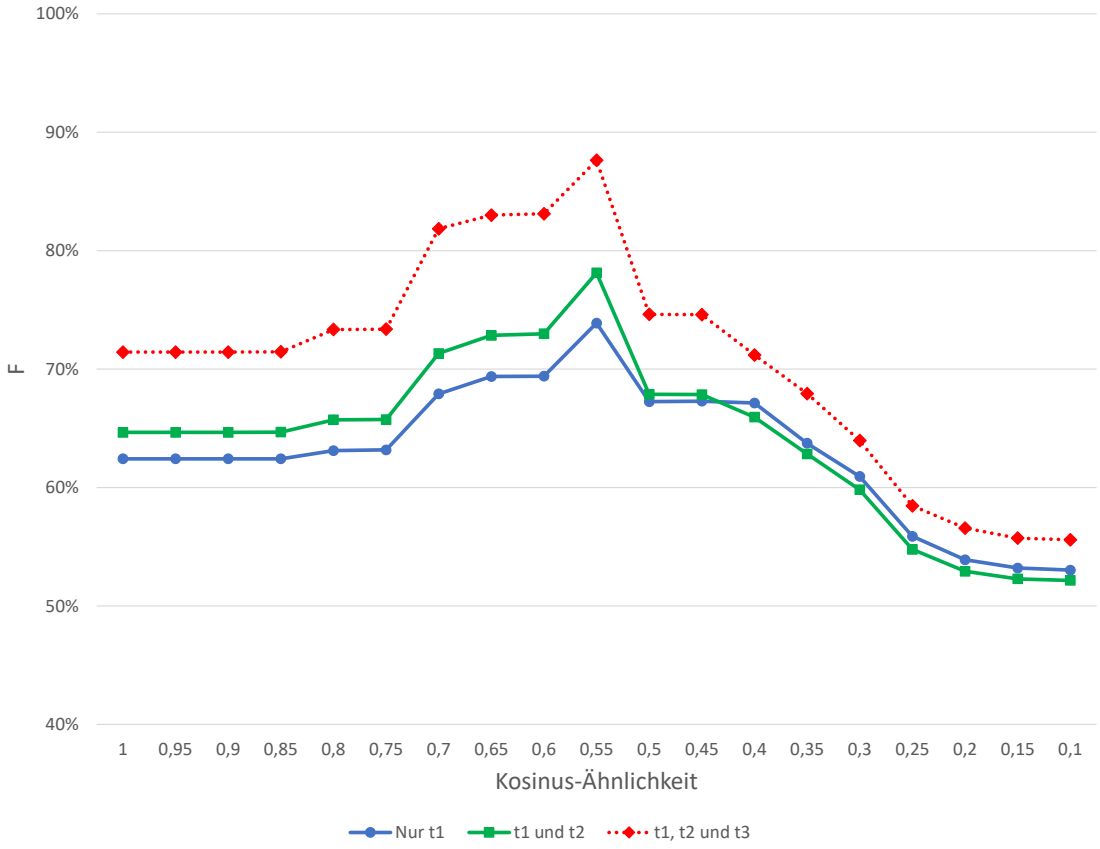

Abbildung 7.2.: Durchschnittliche Werte für das F-Maß bei Anwendung unterschiedlicher Tokenisierungsregeln für alle 16 analysierten Projekte bei Verwendung verschiedener Kosinus-Ähnlichkeitsschwellwerte im Bereich 1 bis 0,1.

Tabelle 7.3.: Durchschnittliche Werte für Precision, Recall und F-Maß für den Kosinus-Ähnlichkeitsschwellwert 0,55 unter Anwendung verschiedener Tokenisierungsregeln.

Regel(n)	Precision	Recall	F
t_1	0,96	0,69	0,74
t_1 und t_2	0,96	0,77	0,78
t_1, t_2 und t_3	0,93	0,86	0,88

den Kosinus-Ähnlichkeitsschwellwert 0,55 zeigt. Spalte 1 listet auf, welche Tokenisierungsregeln zur Erfassung der Trace Links in den 16 rekonstruierten Interaction-Room-Projekten verwendet wurden. In den Spalten 2 bis 4 ist daneben aufgeführt, welche durchschnittlichen Werte für Precision, Recall und F sich dabei jeweils ergeben.

Bei alleiniger Verwendung der ersten Tokenisierungsregel t_1 kann eine sehr hohe Precision von 0,96 erreicht werden. Der entsprechende Recall ist jedoch mit 0,69 sehr gering und auch F fällt mit 0,74 nur unwesentlich höher aus.

Durch Hinzunahme der zweiten Tokenisierungsregel t_2 kann die Qualität der erfassten Trace Links bereits wesentlich verbessert werden: Der Recall steigt um 8 Prozentpunkte von 0,69 auf 0,77 und auch der Wert des F-Maßes kann von 0,74 auf 0,78 erhöht werden. Die Precision bleibt hierbei konstant bei 0,96.

Indem auch die dritte Tokenisierungsregel t_3 hinzugenommen wird, können die Werte erneut signifikant verbessert werden: Obwohl die Precision um 0,03 abfällt, kann der Recall um 9 Prozentpunkte von 0,77 auf 0,86 gesteigert werden. Auch F erhöht sich wesentlich um 10 Prozentpunkte von 0,78 auf 0,88.

Insgesamt wird somit deutlich, dass eine kombinierte Anwendung aller drei Tokenisierungsregeln t_1, t_2 und t_3 sinnvoll ist und mit Abstand die besten Ergebnisse bei der Trace-Link-Erfassung erzielt. Werden nur eine oder zwei dieser Regeln verwendet, so fällt das Ergebnis durchschnittlich wesentlich schlechter aus.

7.3.3. Entfernung von Stoppwörtern

Wörter, die in einer Menge betrachteter Dokumente zu häufig vorkommen und dadurch in fast jedem Dokument enthalten sind, eignen sich im Allgemeinen nicht gut als Suchterme für die Erfassung von Trace Links. Sie besitzen aufgrund ihrer inflationären Verwendung oftmals keine Relevanz für die Analyse der Inhalte und werden deshalb normalerweise herausgefiltert. Im Information Retrieval bezeichnet man solche Wörter als *Stoppwörter* [13]. Typische Beispiele hierfür sind Artikel, Präpositionen und Konjunktionen, weil sie in der Regel lediglich eine grammatikalische Funktion haben, aber keine Rückschlüsse auf die Inhalte eines Dokuments zulassen [71].

Insbesondere für umfangreiche Dokumente ist eine Entfernung von Stoppwörtern häufig sinnvoll, weil sie oft einen großen Teil der Texte ausmachen. Studien haben gezeigt, dass hierdurch der Umfang der Texte um bis zu 40 % reduziert werden kann [13]. Dies wirkt sich in naheliegender Weise auf die Geschwindigkeit der Trace-Link-Erfassung aus, weil die resultierenden Häufigkeitsvektoren und die zugrundeliegende Gesamtmenge aller Wörter dadurch wesentlich verkleinert werden.

Obwohl es keine einheitliche und standardisierte Stoppwort-Liste gibt, existieren zahlreiche populäre Listen, die frei zugänglich sind und häufig verwendet werden. Fox [75] präsentiert beispielsweise eine Stoppwort-Liste für die englische Sprache, die aus der Analyse allgemeiner englischer Literatur entstanden ist und 421 Stoppwörter umfasst. Eine umfangreichere Stoppwort-Liste mit 571 englischen Wörtern wird von Salton und Buckley [33] vorgeschlagen. Auch für die deutsche Sprache existieren Stoppwort-Listen, wie beispielsweise die Liste von Götze und Geyer [83] mit 598 Wörtern.

Trotz der vermeintlichen Vorteile kommt im AugIR aus zwei Gründen keine Entfernung von Stoppwörtern zum Einsatz:

Zum einen sind die betrachteten Dokumente im Interaction-Room-Kontext wesentlich kürzer, als dies ansonsten vielfach im Information Retrieval üblich ist. Der Geschwindigkeitsvorteil, der sich normalerweise aus der Entfernung von Stoppwörtern ergibt, ist im AugIR deshalb vernachlässigbar.

Tabelle 7.4.: Durchschnittliche Werte für Precision, Recall und F-Maß für den Kosinus-Ähnlichkeitsschwellwert 0,55 bei Entfernung bzw. Nichtentfernung von Stoppwörtern.

Entfernung von Stoppwörtern	Precision	Recall	F
ja	0,88	0,85	0,86
nein	0,93	0,86	0,88

Zum anderen hat ein Experiment gezeigt, dass durch eine Stoppwort-Entfernung sowohl Precision als auch Recall bei der Erfassung von Trace Links in den digital rekonstruierten Interaction-Room-Projekten sinken.

Hierzu wurden sämtliche Stoppwörter aus allen erkannten Handschrifttexten vor Durchführung der Trace-Link-Erfassung entfernt und das Ergebnis wurde mit den Resultaten aus Abschnitt 7.3.1 verglichen. Für das einzige englische Projekt P_1 kam dabei die Stoppwort-Liste von Salton und Buckley [33] zum Einsatz, während für alle anderen Projekte $P_2, \ldots, P_{16}$ die Stoppwort-Liste von Götze und Geyer [83] verwendet wurde.

Tabelle 7.4 zeigt die durchschnittlichen Werte für Precision, Recall und F bei Erfassung von Trace Links mit und ohne Entfernung von Stoppwörtern. Man sieht sofort, dass alle drei Werte bei Nichtentfernung der Stoppwörter höher sind. Wenn vor der Trace-Link-Erfassung Stoppwörter entfernt werden, so verringert dies die Precision um 5 Prozentpunkte von 0,93 auf 0,88. Der Recall sinkt hierbei von 0,86 auf 0,85 und auch der Wert des F-Maßes fällt von 0,88 auf 0,86 ab.

Der Grund für die höhere Genauigkeit und Trefferquote bei Berücksichtigung aller Wörter (inklusive Stoppwörtern) liegt darin, dass die Texte in den betrachteten Interaction-Room-Skizzen wie zuvor erläutert oft verhältnismäßig kurz sind. Die Texte werden von den Stakeholdern in der Regel bewusst knapp gehalten oder stichpunktartig formuliert, weshalb Stoppwörter darin nicht übermäßig häufig vorkommen. Hierdurch haben Stoppwörter zwar einen deutlich größeren Einfluss auf das Ergebnis, sie sind jedoch auch oftmals relevant, wenn sie in einem Text erscheinen.

Es ist deshalb sinnvoll, die tokenisierten erkannten Handschrifttexte ungefiltert zur Erfassung von Trace Links zu verwenden, ohne zuvor etwaige Stoppwörter zu entfernen.

7.4. Gefahren für die Gültigkeit

In der vorgestellten Studie wurden 15 Projekte mit deutschen Texten und nur ein Projekt mit englischen Texten analysiert. Da die erfassten Trace Links auf Text-Operationen basieren, kann das ermittelte Ergebnis deshalb nicht ohne Weiteres pauschal auf Projekte beliebiger Sprachen verallgemeinert werden.

Es konnte gezeigt werden, dass eine gleichzeitige Anwendung der drei identifizierten Tokenisierungsregeln t_1, t_2 und t_3 die besten Ergebnisse liefert. Obwohl diese Regeln aus einer gründlichen Analyse der rekonstruierten Projekte entstanden sind, können sie keinen Anspruch auf Vollständigkeit erheben. Es ist nicht ausgeschlossen, dass die Verwendung zusätzlicher Regeln die Qualität der Trace-Link-Erfassung weiter steigern kann. Im Hinblick darauf wurde der Traceability-Algorithmus bewusst generisch definiert, so dass eine Hinzunahme weiterer Regeln oder eine Modifikation bereits existierender Regeln problemlos möglich ist.

Darüber hinaus wurden die korrekten Trace Links für jedes Projekt im Vorfeld von drei unabhängigen Softwareingenieuren geschätzt und in entsprechenden Referenzlisten festgehalten. Obwohl alle drei eine umfassende Kenntnis der Interaction-Room-Methode besitzen und mit den Projektdetails vertraut sind, hängen die geschätzten Trace Links somit zum Teil von Erfahrung, Domänenwissen und tatsächlichem Projektverständnis dieser Experten ab. Es ist deshalb nicht auszuschließen, dass andere Personen geringfügig andere Trace Links schätzen würden. Dies würde zu einer entsprechenden Abweichung in den Resultaten hinsichtlich Precision und Recall führen.

Die rekonstruierten Projekte wurden sowohl zur Kalibrierung als auch zur Evaluation der vorgestellten Verfahren verwendet. Es ist deshalb möglich, dass die Algorithmen einen Bias gegenüber den verwendeten Datensätzen aufweisen und somit

insbesondere für diese Projekte aber nicht notwendigerweise für alle theoretisch möglichen Projekte die besten Ergebnisse erzielen.

Um dieser Gefahr entgegenzuwirken, wurden zum einen möglichst heterogene Projekte verwendet, die eine große Bandbreite möglicher Domänen und Projektsituationen repräsentieren und überdies aus realen Industrieprojekten stammen, die mit der Interaction-Room-Methode unterstützt wurden. Zum anderen wurden diese Projekte von verschiedenen Personen mit unterschiedlichen Handschriften rekonstruiert, um reale Workshop-Szenarien zu emulieren, in denen die Skizzen typischerweise ebenfalls von verschiedenen Stakeholdern gezeichnet werden. Es kann somit angenommen werden, dass die Ergebnisse der Studie mindestens auf zukünftige Interaction-Room-Projekte in deutscher Sprache übertragbar sind.

7.5. Zusammenfassung und Diskussion

In diesem Kapitel konnte gezeigt werden, dass der in Abschnitt 3.8 vorgestellte Algorithmus zur Erfassung von Trace Links bei der Analyse der 16 digital rekonstruierten Interaction-Room-Projekte sehr gute Resultate liefert. Bei Verwendung eines Kosinus-Ähnlichkeitsschwellwerts von $\nu := 0{,}55$ kann durchschnittlich eine Precision von 93 % und ein Recall von 86 % erreicht werden. Der entsprechende Wert des F-Maßes beträgt hierbei 0,88.

Im Durchschnitt wurden für jedes Projekt 43,88 Trace Links gefunden, wovon 40,69 korrekt sind. Pro Projekt existieren durchschnittlich lediglich 3,19 False Positives und 4,5 übersehene Trace Links. Für zehn der 16 Projekte konnte für $\nu = 0{,}55$ somit ein Recall-Wert ≥ 90 % erreicht werden. Lediglich drei Projekte weisen einen Recall unter 80 % auf. Eine Precision ≥ 90 % wurde sogar für 13 der 16 Projekte erzielt, wobei nur für ein einziges Projekt eine Precision unter 80 % beobachtet wurde.

Zusammenfassend lässt sich somit festhalten, dass der vorgestellte Algorithmus in der Lage ist, die meisten relevanten Trace Links in fast jedem untersuchten Projekt

aufzudecken und dabei nur eine akzeptabel kleine Anzahl falsch positiver Treffer produziert.

Weiterhin wurde in diesem Kapitel deutlich, dass eine kombinierte Anwendung der Tokenisierungsregeln t_1, t_2 und t_3 auf die erkannten Handschrifttexte vor Erfassung der Trace Links die Qualität des Algorithmus signifikant verbessern kann. Werden nur eine oder zwei dieser Regeln verwendet, so fällt das Ergebnis durchschnittlich wesentlich schlechter aus.

Obwohl die identifizierten Tokenisierungsregeln keinen Anspruch auf Vollständigkeit erheben, ist anzunehmen, dass die erzielten Ergebnisse durch Hinzunahme weiterer Regeln nur schwer verbessert werden können. Zum einen sind bestimmte Handschrifterkennungsfehler so gravierend, dass sie weder durch Anwendung zusätzlicher Regeln noch durch eine unscharfe Suche korrigiert werden können. Zum anderen existieren bestimmte Trace Links nicht aufgrund textueller Ähnlichkeiten sondern lediglich aufgrund semantischer Beziehungen. Solche Trace Links können von einem retrospektiven Traceability-Ansatz grundsätzlich nicht erfasst werden.

Abschließend wurde gezeigt, dass eine vorherige Entfernung von Stoppwörtern nicht sinnvoll ist, weil hierdurch Precision und Recall für die analysierten Interaction-Room-Projekte sinken. Dies kommt daher, dass die Texte in den Skizzen häufig kurz sind und Stoppwörter darin nicht übermäßig oft vorkommen. Hierdurch besitzen Stoppwörter oftmals eine Relevanz, wenn sie in einem Text erscheinen. Die erkannten Handschrifttexte werden deshalb ungefiltert zur Erfassung von Trace Links verwendet, ohne zuvor etwaige Stoppwörter zu eliminieren.

8. Identifikation inhaltlicher Zusammenhänge

Nachdem im vorherigen Kapitel mit mathematischen Methoden die Qualität der erfassten Trace Links analysiert wurde, soll nun ihre tatsächliche Nützlichkeit für die Stakeholder im praktischen Einsatz gezeigt werden. Hierzu wird im Folgenden im Rahmen eines umfangreichen Experiments untersucht, ob Trace Links im AugIR dabei helfen können, inhaltliche Zusammenhänge in Projekten schneller und zuverlässiger zu erkennen, als dies auf analogen Whiteboards möglich ist. Dem Experiment liegen dabei die beiden folgenden Forschungshypothesen zugrunde:

1. Stakeholder können inhaltliche Zusammenhänge zwischen Skizzen im AugIR schneller erkennen als auf analogen Whiteboards.

2. Stakeholder übersehen im AugIR seltener inhaltliche Zusammenhänge als auf analogen Whiteboards.

Im Folgenden wird zunächst der grundlegende Aufbau des Experiments erläutert. Danach wird diskutiert, nach welchen Gesichtspunkten geeignete Projekte für die Durchführung ausgewählt wurden. Anschließend wird die zugrundeliegende Aufgabenstellung beschrieben und es wird geschildert, welche Messdaten im Laufe des Experiments erhoben wurden. Danach werden die ermittelten Ergebnisse präsentiert. Im Anschluss daran werden Fragebögen ausgewertet, die nach dem Experiment von den Teilnehmern ausgefüllt wurden. Das Kapitel schließt mit einer Betrachtung der Gefahren für die Gültigkeit sowie mit einer Zusammenfassung und Diskussion der gewonnenen Erkenntnisse.

8.1. Versuchsaufbau und Durchführung

Zur Teilnahme am folgenden Experiment wurden 22 Personen eingeladen. Jedem Teilnehmer des Experiments wurden mehrere Aufgaben gestellt. Diese waren der Reihe nach in Einzelarbeit zu lösen, um eine gegenseitige Beeinflussung der Teilnehmer auszuschließen. Die eine Hälfte der Aufgaben war auf analogen Whiteboards im Interaction Room zu bearbeiten, die andere Hälfte war auf digitalen Displays mit Hilfe des AugIRs zu lösen.

Alle Aufgaben sollten anhand realer Interaction-Room-Projekte bearbeitet werden, damit die erzielten Ergebnisse auf zukünftige Projekte übertragbar sind. Hierzu wurden zwei geeignete Projekte aus der Gesamtheit aller in Abschnitt 3.3 vorgestellten rekonstruierten Projekte ausgewählt. Der detaillierte Auswahlprozess wird in Abschnitt 8.2 beschrieben.

Um einem möglicherweise auftretenden Lerneffekt entgegenzuwirken, wurden alle Teilnehmer des Experiments wie in Tabelle 8.1 dargestellt in zwei Gruppen eingeteilt: Teilnehmer der ersten Gruppe hatten zunächst in Runde 1 zwei Aufgaben in Projekt 1 im Interaction Room zu bearbeiten. Danach mussten sie in Runde 2 die beiden gleichen Aufgaben erneut in Projekt 2 im AugIR bearbeiten. Für Teilnehmer der zweiten Gruppe war die Reihenfolge genau umgekehrt. Sie mussten zunächst in Runde 1 zwei Aufgaben in Projekt 1 im AugIR lösen und danach die gleichen Aufgaben in Runde 2 in Projekt 2 im Interaction Room. Dem Versuchsaufbau liegt somit ein 2-faktorielles Design zugrunde.

Die Bearbeitung im Interaction Room erfolgte dabei konventionell auf analogen Whiteboards ohne den Einsatz technischer Hilfsmittel. Bei der Bearbeitung im AugIR durften die Teilnehmer die digitalen Displays verwenden und wurden von der AugIR-Software bei der Lösung der Aufgaben unterstützt.

Die 22 Teilnehmer im Alter von 21 bis 42 Jahren hatten unterschiedliche Hintergründe und Kenntnisstände. Die Gruppe setzte sich aus drei weiblichen und 19 männlichen Probanden zusammen. 15 Personen waren wissenschaftliche Mitarbeiter der Universität und verfügten über einen Studienabschluss in Informatik oder verwandten Gebieten. Sieben Personen waren Studenten der Angewandten Informatik

Tabelle 8.1.: Einteilung der Probanden in zwei Versuchsgruppen.

	Runde 1		Runde 2	
Gruppe 1	Projekt 1 Aufgabe 1 IR	Projekt 1 Aufgabe 2 IR	Projekt 2 Aufgabe 1 AugIR	Projekt 2 Aufgabe 2 AugIR
Gruppe 2	Projekt 1 Aufgabe 1 AugIR	Projekt 1 Aufgabe 2 AugIR	Projekt 2 Aufgabe 1 IR	Projekt 2 Aufgabe 2 IR

oder Wirtschaftsinformatik. Alle Teilnehmer hatten Kenntnisse in mindestens einer Modellierungssprache wie UML oder BPMN. Zwei Teilnehmer waren erfahrene Interaction-Room-Coaches, die bereits mehrere Interaction-Room-Workshops geleitet haben. Zehn Teilnehmer hatten zuvor keinerlei Erfahrungen mit der Interaction-Room-Methode, während die restlichen Teilnehmer über rudimentäre Grundkenntnisse verfügten.

Der Ablauf des Experiments war wie folgt:

Um alle Teilnehmer auf den gleichen Wissensstand zu bringen, startete jeder Versuchsdurchlauf zunächst mit einer fünfminütigen Einführung in die Interaction-Room-Methode. Während dieser Zeit erläuterte der Moderator des Experiments dem Teilnehmer die Grundlagen des Interaction Rooms. Es wurden die für das Experiment wesentlichen Canvas-Typen sowie der Zweck von Annotationen erklärt.

Darauf folgte eine fünfminütige Erläuterung der Aufgabenstellung. Diese wird im Detail in Abschnitt 8.3 vorgestellt.

Anschließend erhielt jeder Teilnehmer eine kurze Einführung in die AugIR-Software. Hierzu erläuterte der Moderator zunächst die wesentlichen Funktionalitäten der Software und demonstrierte deren Verwendung an einem realen Beispielprojekt. Anschließend durfte der Proband die Software frei ausprobieren und sich mit deren Benutzung vertraut machen.

Danach startete die Bearbeitung der Aufgaben in zwei Runden. Wie eingangs beschrieben bearbeiteten Teilnehmer der ersten Gruppe die Aufgaben in Runde 1 im Interaction Room und in Runde 2 im AugIR, während Teilnehmer der zweiten Gruppe die Aufgaben in Runde 1 im AugIR und danach in Runde 2 im Interaction Room bearbeiteten.

Abschließend füllte jeder Teilnehmer der Studie einen Fragebogen aus. Um die Teilnehmer hierbei nicht zu beeinflussen, geschah das Ausfüllen vollkommen anonym. Alle Fragebögen wurden hierzu ohne Aufsicht ausgefüllt, anschließend zunächst in einer verschlossenen Box gesammelt und erst nach Beendigung des Gesamtexperiments ausgewertet.

8.2. Auswahl der Projekte

Im Folgenden soll erläutert werden, anhand welcher Gesichtspunkte zwei geeignete Projekte für die Durchführung des Experiments ausgewählt wurden. Hierzu wurden drei Kriterien betrachtet:

Kriterium 1: Komplexität der Projekte

Die Projekte sollten weder zu groß noch zu klein sein, um die Schwierigkeit der zu bearbeitenden Aufgaben weder positiv noch negativ zu beeinflussen. Die verwendeten Canvas-Typen sollten in Art und Anzahl einem durchschnittlichen Interaction-Room-Projekt entsprechen.

Die wesentlichen Eigenschaften der 16 zur Verfügung stehenden Projekte sind im Detail aus Abschnitt 3.3 ersichtlich. Zusammenfassend lässt sich festhalten, dass diese durchschnittlich jeweils meist über einen Feature Canvas, drei bis vier Process Canvases und einen Object Canvas verfügen. Ein Integration Canvas ist nur in weniger als der Hälfte der Projekte enthalten.

Durchschnittlich wurden in jedem Projekt etwa fünf bis sechs Skizzen erstellt, die insgesamt etwa 125 Textelemente enthalten. Davon befinden sich im Durchschnitt

etwa 22 % der Elemente auf dem Feature Canvas, 50 % auf den Process Canvases und 16 % auf dem Object Canvas. Falls ein Integration Canvas vorhanden ist, so entfallen auf diesen die restlichen 12 % der Elemente. Andernfalls verteilen sie sich auf die anderen Canvases.

Kriterium 2: Qualität der Trace Links

Im Rahmen dieses Experiments soll analysiert werden, in wie fern Trace Links im AugIR die Stakeholder bei ihrer Arbeit unterstützen können. Projekte mit besonders guten oder besonders schlechten Werten bei der Trace-Link-Erfassung würden an dieser Stelle deshalb möglicherweise das Ergebnis verzerren.

Wie in Kapitel 7 diskutiert, wurde $\nu := 0{,}55$ als Schwellwert für die Betrachtung der Kosinus-Ähnlichkeit gewählt. Hierdurch können durchschnittlich in jedem Projekt ca. 44 Trace Links erfasst werden, wovon ca. 41 korrekt sind. Dies führt im Durchschnitt zu einer Precision von 93 % und einem Recall von 86 %.

Kriterium 3: Allgemeinverständlichkeit

Um die Schwierigkeit der gestellten Aufgaben nicht zu verzerren, sollten darüber hinaus zwei Projekte gewählt werden, die inhaltlich soweit wie möglich allgemeinverständlich sind und kein spezifisches Domänenwissen erfordern. Es wurde deshalb darauf geachtet, dass die ausgewählten Projekte inhaltlich möglichst leicht verständlich und für jedermann nachvollziehbar sind. Sofern projekt- und domänenspezifische Fachbegriffe vorkommen, müssen sich diese für jeden Teilnehmer leicht und intuitiv aus dem Kontext erschließen lassen.

Bewertung der Projekte

Betrachtet man die Merkmale aller 16 Projekte, so stellen sich P_5 und P_{12} als die am besten geeignetsten Kandidaten heraus:

Projekt P_5 enthält einen Feature Canvas, vier Process Canvases und einen Object Canvas. Dies entspricht genau den durchschnittlichen Werten. Das Projekt liegt mit insgesamt 95 Textelementen zwar leicht unter der durchschnittlichen Projektgröße, was jedoch das Ergebnis des Experiments nicht zugunsten des AugIRs beeinflusst, weil hierdurch die Schwierigkeit der zu lösenden Aufgaben für die Probanden im Interaction Room höchstens gesenkt aber nicht gesteigert wird. Darüber hinaus passt die Verteilung der Elemente auf die einzelnen Landkarten, wenn man berücksichtigt, dass dieses Projekt keinen Integration Canvas enthält und der prozentuale Anteil dieser Elemente sich deshalb auf Feature und Object Canvas verteilt. Der Traceability-Algorithmus erfasst in Projekt P_5 für den Schwellwert $\nu = 0{,}55$ insgesamt 49 Trace Links, von denen 45 korrekt sind. Es gibt vier False Positives und fünf übersehene Trace Links. Dies resultiert in einer Precision von 92 % und einem Recall von 90 %. Diese Werte entsprechen ebenfalls sehr gut dem zuvor ermittelten Durchschnitt. Auch inhaltlich ist Projekt P_5 für das Experiment geeignet, weil die skizzierten Inhalte bis auf wenige Ausnahmen allgemeinverständlich sind und kein spezifisches Domänen- oder Fachwissen erfordern.

Projekt P_{12} enthält 136 Elemente, die sich auf einen Feature Canvas, sechs Process Canvases und einen Object Canvas verteilen. Damit liegt dieses Projekt hinsichtlich seines Umfangs nur geringfügig über dem Durchschnitt und die Verteilung der Elemente auf die einzelnen Landkarten passt gut zu den ermittelten Durchschnittswerten. Der Traceability-Algorithmus erfasst 33 Trace Links, von denen 31 korrekt sind. Es gibt zwei False Positives und einen übersehenen Trace Link. Dies führt zu einer Precision von 94 % und einem Recall von 97 %. Obwohl der Recall somit über dem Durchschnitt liegt, existiert kein anderes Projekt, das einen geeigneteren Recall aufweist und dennoch alle anderen geforderten Eigenschaften gleichermaßen gut erfüllt. P_{12} ist außerdem sehr leicht verständlich, weil es keinerlei domänen- oder projektspezifische Fachbegriffe enthält. Die skizzierten Abläufe können problemlos von jedem Betrachter nachvollzogen werden, was die Schwierigkeit der zu lösenden Aufgaben insbesondere für die Probanden im Interaction Room senkt und die geringfügig überdurchschnittliche Trefferquote im AugIR teilweise ausgleicht.

8.3. Aufgabenstellung

Wie eingangs erläutert soll in diesem Experiment überprüft werden, wie gut die erfassten Trace Links die Stakeholder beim Erkennen von Zusammenhängen unterstützen. Hierzu wurde jeder Teilnehmer des Experiments gebeten, in den ausgewählten Projekten P_{12} und P_5 jeweils der Reihe nach zwei Aufgaben zu bearbeiten.

Die Reihenfolge der Projekte wurde hierbei nicht alterniert: Beide Gruppen bearbeiteten Projekt P_{12} in Runde 1 und Projekt P_5 in Runde 2. Die Reihenfolge der Projekte wurde unter Berücksichtigung der Tatsache gewählt, dass Projekt P_{12} inhaltlich etwas leichter verständlich ist als Projekt P_5. Dies ermöglichte den Teilnehmern beider Gruppen einen sanfteren Einstieg in das Experiment.

Die erste Aufgabe bestand darin, im vorliegenden Projekt Zusammenhänge zwischen dem Feature Canvas und den Process Canvases zu identifizieren. Stakeholder sollten hierzu ermitteln, welche Features des Feature Canvas durch welche Prozessschritte auf den Process Canvases abgebildet oder referenziert werden. Die entsprechenden Features sowie die zugehörigen Prozessschritte waren zu identifizieren und deutlich zu nennen.

Alle von einem Teilnehmer gefundenen Zusammenhänge wurden während des Experiments von einem Schriftführer protokolliert. Auf diese Weise konnten sich die Teilnehmer vollständig auf die Durchführung der Aufgabe konzentrieren und haben keine Zeit beim Aufschreiben der Ergebnisse verloren, was ansonsten die Resultate hätte verfälschen können.

Alle Zusammenhänge, die es in der ersten Aufgabe zu finden galt, waren offensichtlich und nicht versteckt. Es handelte sich dabei ausdrücklich nicht um semantische Zusammenhänge, weil diese von den Teilnehmern der Studie in der Regel ohne umfassendes Projekt- und Domänenwissen nicht hätten aufgedeckt werden können. Alle gesuchten Beziehungen konnten mit einer rein inhaltlichen Analyse der Texte gefunden werden.

Beispielsweise bezog sich eine Aktivität „Vertrag verarbeiten“ auf einem Process Canvas auf ein Feature „Vertragsverarbeitung“ auf einem Feature Canvas. Entspre-

chend wurde ein Feature „Erstellung der Zahlung“ auf einem Feature Canvas durch eine Aktivität „Zahlung erstellen“ auf einem Process Canvas abgebildet. Beide Zusammenhänge können durch eine gründliche Betrachtung der Canvases aufgedeckt werden, ohne dass hierfür Kenntnisse des Projekts oder der Domäne notwendig sind.

Demgegenüber existieren in fast allen Projekten aber auch solche Zusammenhänge, die sich nur mit ausreichendem Detailwissen offenbaren. Beispielsweise kann sich ein fachliches Objekt „Auftrag“ in einem gegebenen Projektkontext auf eine Bestellung beziehen, so dass eine Beziehung zu einer Aktivität „Bestellung erfassen“ existieren würde. Ein solcher Zusammenhang lässt sich aber nur mit ausreichendem Projektwissen erkennen, so dass eine Identifikation dieser Art von Beziehungen im vorliegenden Experiment explizit nicht gefordert wurde.

Die erste Aufgabe galt als abgeschlossen, sobald der Teilnehmer alle existierenden Zusammenhänge aufgedeckt hatte. Er musste hierfür alle Features des Feature Canvas der Reihe nach betrachten und jeweils eventuell vorhandene Zusammenhänge finden. Übersah der Teilnehmer einen Zusammenhang für ein Feature, so wurde ihm dies vom Moderator mitgeteilt, ohne dass ihm die genaue Anzahl der übersehenen Zusammenhänge oder deren Position verraten wurden. Daraufhin musste er die Suche fortsetzen, bis er die noch fehlende(n) Beziehung(en) für das entsprechende Feature aufdecken konnte. Es war somit eindeutig mess- und entscheidbar, wann die Aufgabe für den jeweiligen Teilnehmer abgeschlossen war.

Die zweite Aufgabe bestand darin, mehrfach vorkommende Annotationsnummern zu identifizieren. Wie in Abschnitt 1.2 erläutert, wird jede Annotation mit einer Nummer versehen, um diese eindeutig einer zugehörigen Erläuterung zuzuordnen. Wenn im Verlauf eines Interaction-Room-Workshops Stakeholder zu unterschiedlichen Annotationen die gleichen Erläuterungen geben, so erhalten diese Annotationen in naheliegender Weise die gleichen Nummern, um identische Aussagen nicht mehrfach notieren zu müssen. Die Aufgabe der Stakeholder war es, für jede im Feature Canvas vorkommende Annotation zu untersuchen, ob andere Canvases Annotationen mit der gleichen Nummer enthalten. Die entsprechenden Annotationen waren dem Moderator des Experiments zu nennen und wurden vom Schriftführer protokol-

liert. Die zweite Aufgabe galt dementsprechend als abgeschlossen, wenn der jeweilige Teilnehmer alle Vorkommnisse einer Annotationsnummer in allen Skizzen identifiziert und genannt hatte. Übersah der Teilnehmer eine Annotation, so wurde ihm dies vom Moderator mitgeteilt, ohne dass ihm die genaue Anzahl der übersehenen Annotationen oder deren Position verraten wurden. Daraufhin musste er die Suche fortsetzen, bis er die noch fehlende(n) Annotation(en) aufdecken konnte.

8.4. Messdatenerhebung

Im Rahmen dieses Experiments wurden bei der Durchführung der beiden Aufgaben für jeden Teilnehmer in jeder Runde jeweils vier Messwerte erhoben:

Für die erste Aufgabe wurde gemessen, wie lange ein Teilnehmer brauchte, um alle Zusammenhänge für alle Features des Feature Canvas zu identifizieren und korrekt zu nennen. Zum anderen wurde protokolliert, wie viele Fehler er dabei machte. Ein Fehler wurde jedes Mal dann notiert, wenn ein Teilnehmer glaubte, für ein betrachtetes Feature alle Zusammenhänge aufgedeckt zu haben, es aber noch weitere übersehene Zusammenhänge gab.

Für die zweite Aufgabe wurden erneut die beiden vorgenannten Messwerte erhoben: Es wurde gemessen, wie lange ein Teilnehmer brauchte, um alle mehrfach vorkommenden Annotationsnummern zu finden. Darüber hinaus wurde die Anzahl der hierbei gemachten Fehler protokolliert. Analog zu Aufgabe 1 wurde jedes Mal dann ein Fehler notiert, wenn ein Teilnehmer fälschlicherweise glaubte, alle Vorkommnisse einer bestimmten Annotationsnummer gefunden zu haben, diese aber noch an weiteren von ihm übersehenen Stellen vorkam.

8.5. Ergebnisse

Jeder Teilnehmer wurde gebeten, die zuvor beschriebenen Aufgaben nacheinander in beiden Projekten P_{12} und P_5 durchzuführen. Wie eingangs erläutert entschied die

Tabelle 8.2.: Zeiten und Fehler für Teilnehmer der ersten Gruppe.

	Runde 1 Interaction Room				Runde 2 AugIR			
	Aufgabe 1		Aufgabe 2		Aufgabe 1		Aufgabe 2	
Teilnehmer	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler
T_1	26:09	6	15:44	3	06:37	0	02:50	0
T_2	26:05	1	15:37	2	05:49	0	03:51	0
T_3	19:41	3	13:29	3	04:50	0	01:54	0
T_4	18:09	5	17:39	2	04:52	0	01:44	0
T_5	24:09	4	18:55	1	05:41	0	02:30	0
T_6	30:14	9	16:59	5	09:56	0	02:39	0
T_7	20:31	5	17:29	3	05:23	0	02:22	0
T_8	22:19	7	07:55	1	03:45	0	01:39	0
T_9	29:04	7	17:08	3	05:48	0	02:28	0
T_{10}	25:24	6	19:48	5	06:41	0	02:20	0
T_{11}	21:42	6	15:21	4	06:27	0	02:09	0
Ø	23:57	5,36	16:00	2,91	05:59	0	02:24	0

Gruppenzugehörigkeit der Teilnehmer darüber, in welcher Runde jeweils die Aufgaben auf analogen Whiteboards im Interaction Room oder auf digitalen Whiteboards im AugIR zu bearbeiten waren.

Im Folgenden werden die Resultate beider Gruppen zunächst nacheinander vorgestellt, bevor sie schließlich zusammengefasst und bewertet werden.

8.5.1. Resultate für Gruppe 1

Tabelle 8.2 zeigt die Resultate des Experiments für Gruppe 1.

In der ersten Spalte ist das Kürzel des jeweiligen Teilnehmers eingetragen, dessen Messwerte in der entsprechenden Zeile daneben aufgeführt sind. Zur Anonymisierung werden alle Teilnehmer der ersten Gruppe durch ein Kürzel $T_1, \ldots, T_{11}$ identifiziert.

Spalte 2 enthält die Zeit in Minuten und Sekunden im Format „mm:ss“, die der entsprechende Teilnehmer zur vollständigen und korrekten Lösung der ersten Aufgabe in Projekt P_{12} im Interaction Room benötigte. Spalte 3 zeigt die Anzahl der Fehler, die er dabei machte. In den Spalten 4 und 5 sind entsprechend die benötigte Zeit und die Anzahl der Fehler für die Lösung der zweiten Aufgabe in Projekt P_{12} im Interaction Room aufgeführt.

In gleicher Weise zeigen die Spalten 6 bis 9 die benötigten Zeiten und gemachten Fehler, die für den entsprechenden Teilnehmer bei der Bearbeitung der Aufgaben in Projekt P_5 im AugIR protokolliert wurden.

In der letzten Zeile der Tabelle sind darüber hinaus die durchschnittlichen Zeiten und Fehler pro Teilnehmer aufgeführt.

Die durchschnittlich benötigte Zeit zur Lösung der ersten Aufgabe im Interaction Room betrug 23 Minuten und 57 Sekunden. Während der schnellste Teilnehmer T_4 die Aufgabe vollständig in 18 Minuten und 9 Sekunden lösen konnte, benötigte der langsamste Teilnehmer T_6 hierfür 30 Minuten und 14 Sekunden. Eine eventuelle Kenntnis der Interaction-Room-Methode oder Erfahrungen mit grafischen Modellierungssprachen hatten keinen Einfluss auf diese zeitlichen Abweichungen. Weder Teilnehmer T_4 noch Teilnehmer T_6 waren vor dem Experiment mit dem Interaction Room vertraut. Darüber hinaus verfügten beide lediglich über Grundkenntnisse in UML. Trotz dieser zeitlichen Differenzen ist ersichtlich, dass neun von elf Teilnehmern für die Lösung der ersten Aufgabe im Interaction Room mehr als 20 Minuten benötigten.

Dabei gelang es keinem Teilnehmer, die erste Aufgabe im Interaction Room fehlerfrei zu lösen. Die wenigsten Fehler machte Proband T_2, der nur einen einzigen Zusammenhang übersah. Demgegenüber machte Teilnehmer T_6 bei der Lösung der ersten Aufgabe neun Fehler. Durchschnittlich wurden pro Teilnehmer 5,36 Fehler gemacht.

Die zweite Aufgabe im Interaction Room konnte durchschnittlich in 16 Minuten gelöst werden. Hierbei benötigte der langsamste Teilnehmer T_{10} 19 Minuten und 48 Sekunden zur Bearbeitung, während der schnellste Teilnehmer T_8 die Aufgabe in 7

Minuten und 55 Sekunden lösen konnte. Diese beachtlich niedrige Zeit ist darauf zurückzuführen, dass die Teilnehmer explizit beliebige Suchstrategien für das Auffinden der Annotationen anwenden durften. Viele Teilnehmer gingen deshalb dazu über, mehrere Nummern auf einmal zu suchen anstatt eine Annotation nach der anderen zu betrachten. In den meisten Fällen suchten die Teilnehmer zwei oder maximal drei Nummern gleichzeitig. Teilnehmer T_8 gelang es jedoch, zeitweise eine beachtliche Anzahl von sechs Nummern gleichzeitig erfolgreich zu suchen. Dies erklärt die signifikante Zeitersparnis, die dieser Proband bei der Bearbeitung der Aufgabe verzeichnen konnte. Es sei darauf hingewiesen, dass der betreffende Teilnehmer zuvor keinerlei Erfahrung in der Auswertung von Interaction-Room-Projekten besaß.

Trotz seiner beeindruckenden Geschwindigkeit machte Teilnehmer T_8 – genau wie Teilnehmer T_5 – bei der Bearbeitung von Aufgabe 2 nur einen einzigen Fehler. Die meisten Fehler machte Teilnehmer T_6 mit 9 Fehlern. Durchschnittlich wurden pro Teilnehmer 2,91 Fehler gemacht.

In der zweiten Runde des Experiments sank die zur Lösung der ersten Aufgabe benötigte Zeit für alle Teilnehmer, als diese das zweite Projekt mit Unterstützung durch Trace Links im AugIR bearbeiten durften. Die durchschnittliche Bearbeitungszeit betrug lediglich 5 Minuten und 59 Sekunden. Der schnellste Teilnehmer T_8 konnte die erste Aufgabe in 3 Minuten und 45 Sekunden lösen, während der langsamste Teilnehmer T_6 hierfür 9 Minuten und 56 Sekunden benötigte.

Es ist hervorzuheben, dass kein einziger Teilnehmer bei der Bearbeitung der ersten Aufgabe im AugIR einen Fehler machte.

Auch die zweite Aufgabe konnte im AugIR schneller gelöst werden als im Interaction Room. Die durchschnittliche Bearbeitungszeit betrug 2 Minuten und 24 Sekunden. Die Aufgabe konnte erneut von Teilnehmer T_8 innerhalb von 1 Minute und 39 Sekunden am schnellsten gelöst werden. Der langsamste Teilnehmer war Proband T_2 mit 3 Minuten und 51 Sekunden.

Auch bei der Bearbeitung der zweiten Aufgabe im AugIR machte kein Teilnehmer einen Fehler.

Tabelle 8.3.: Zeiten und Fehler für Teilnehmer der zweiten Gruppe.

	Runde 1 AugIR				Runde 2 Interaction Room			
	Aufgabe 1		Aufgabe 2		Aufgabe 1		Aufgabe 2	
Teilnehmer	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler
T_{12}	08:37	0	02:41	0	18:00	6	17:07	8
T_{13}	07:41	0	02:47	0	21:28	1	15:05	2
T_{14}	11:25	0	02:55	0	19:09	2	20:26	1
T_{15}	09:36	0	03:25	0	28:54	2	16:15	3
T_{16}	09:13	0	03:13	0	22:25	0	26:04	1
T_{17}	06:47	0	01:59	0	12:00	3	11:23	4
T_{18}	08:44	0	02:06	0	17:09	3	11:32	4
T_{19}	05:38	0	01:35	0	14:43	5	09:13	3
T_{20}	07:59	0	02:12	0	22:44	1	12:48	2
T_{21}	07:41	0	02:25	0	17:02	3	15:01	4
T_{22}	11:06	0	02:58	0	22:58	5	17:00	7
Ø	08:35	0	02:34	0	19:41	2,82	15:38	3,55

8.5.2. Resultate für Gruppe 2

Die Resultate des Experiments für die zweite Gruppe sind in Tabelle 8.3 dargestellt. Der Aufbau dieser Tabelle entspricht dem zuvor erläuterten Aufbau von Tabelle 8.2. Der wesentliche Unterschied besteht darin, dass die Teilnehmer der zweiten Gruppe in Runde 1 beide Aufgaben im AugIR bearbeiteten und erst in Runde 2 in den Interaction Room wechselten.

Die durchschnittlich benötigte Zeit zur Bearbeitung der ersten Aufgabe im AugIR lag bei 8 Minuten und 35 Sekunden. Teilnehmer T_{19} konnte die Aufgabe am schnellsten lösen und benötigte lediglich 5 Minuten und 38 Sekunden. Der langsamste Teilnehmer war Proband T_{14} – er benötigte 11 Minuten und 25 Sekunden.

Auch in dieser Gruppe konnten alle Teilnehmer bei der Bearbeitung im AugIR alle Zusammenhänge auf Anhieb finden, so dass auch hier die Gesamtzahl der Fehler bei 0 liegt.

Aufgabe 2 konnte im AugIR in durchschnittlich 2 Minuten und 34 Sekunden gelöst werden. Auch bei dieser Aufgabe war Teilnehmer T_{19} am schnellsten und benötigte lediglich 1 Minute und 35 Sekunden. Mit 3 Minuten und 25 Sekunden Bearbeitungszeit war Teilnehmer T_{15} am langsamsten.

Erneut konnten alle Probanden die Aufgabe im AugIR fehlerfrei lösen, ohne dabei einen Zusammenhang zu übersehen. Die Anzahl der Fehler liegt somit auch für Aufgabe 2 bei 0.

Genau wie bei Gruppe 1 benötigten auch die Teilnehmer der zweiten Gruppe für die Bearbeitung der Aufgaben im Interaction Room jeweils länger als im AugIR. Die durchschnittliche Bearbeitungszeit für Aufgabe 1 im Interaction Room in Runde 2 betrug 19 Minuten und 41 Sekunden. Am längsten benötigte Teilnehmer T_{15} zur Lösung der ersten Aufgabe. Er brauchte 28 Minuten und 54 Sekunden, bis er alle Zusammenhänge vollständig erkannt hatte. Der schnellste Teilnehmer war T_{17} mit 14 Minuten und 43 Sekunden.

Die durchschnittliche Fehlerzahl bei der Bearbeitung der ersten Aufgabe im Interaction Room lag bei 2,82. Es gelang nur einem einzigen Teilnehmer, diese Aufgabe im Interaction Room fehlerfrei zu lösen: Teilnehmer T_{16} konnte die Aufgabe in 22 Minuten und 25 Sekunden abschließen, ohne dabei einen inhaltlichen Zusammenhang zu übersehen. Die meisten Fehler machte Teilnehmer T_{12} – er übersah sechs Zusammenhänge.

Aufgabe 2 konnte im Interaction Room in durchschnittlich 15 Minuten und 38 Sekunden gelöst werden. Teilnehmer T_{19} war am schnellsten und benötigte zur vollständigen Bearbeitung der zweiten Aufgabe 9 Minuten und 13 Sekunden, während der langsamste Teilnehmer T_{16} 26 Minuten und 4 Sekunden brauchte.

Keiner der Teilnehmer konnte die zweite Aufgabe im Interaction Room fehlerfrei lösen. Die durchschnittliche Fehleranzahl betrug 3,55. Teilnehmer T_{14} und T_{16} übersahen beide jeweils nur einen Zusammenhang bei der Bearbeitung der Aufgabe, während Teilnehmer T_{12} insgesamt acht Fehler machte.

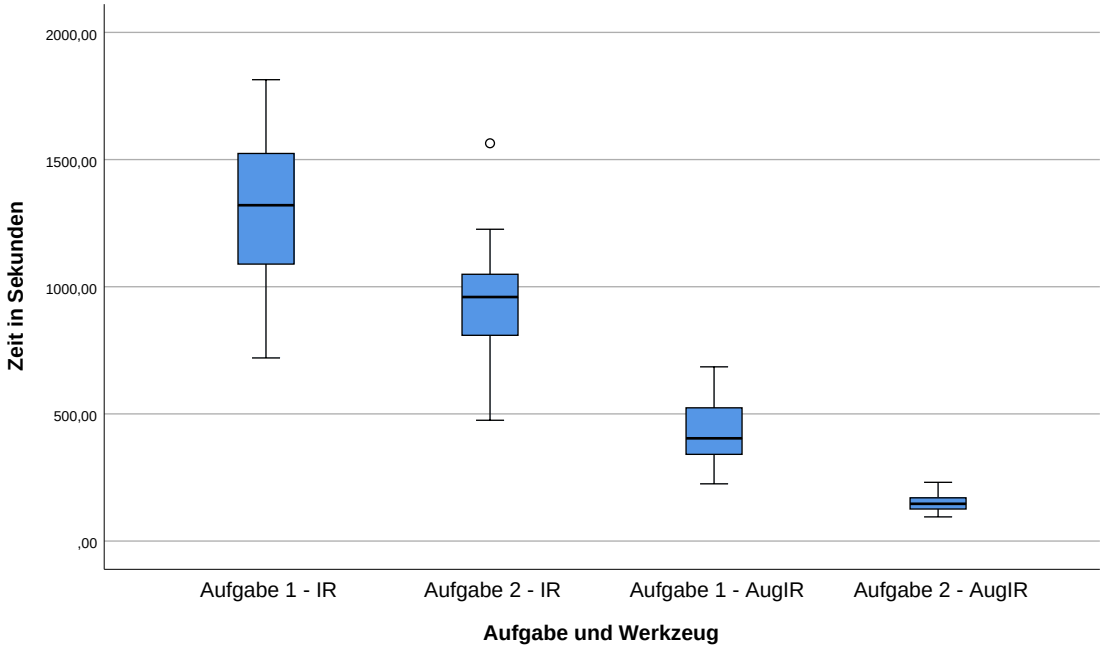

Abbildung 8.1.: Boxplot für die Verteilung der zur Bearbeitung der Aufgaben benötigten Zeiten beider Gruppen.

8.5.3. Zusammenfassung und Bewertung

Abbildung 8.1 visualisiert die Verteilung der zur Bearbeitung der Aufgaben benötigten Zeiten beider Gruppen als Boxplot. Die ersten beiden Boxen stellen die Verteilung der Zeiten dar, die für die Bearbeitung der beiden Aufgaben im Interaction Room benötigt wurden. Entsprechend stellen die dritte und vierte Box die Verteilung der Zeiten dar, die für die Bearbeitung der Aufgaben im AugIR benötigt wurden. Das Diagramm visualisiert noch einmal, dass die Bearbeitung der Aufgaben im Interaction Room jeweils wesentlich länger dauerte als im AugIR. Dabei nahm Aufgabe 1 in beiden Fällen durchschnittlich mehr Zeit in Anspruch als Aufgabe 2.

Entsprechend visualisiert Abbildung 8.2 die Verteilung der bei der Bearbeitung der Aufgaben gemachten Fehler beider Gruppen als Boxplot. Erneut stellen die ersten beiden Boxen die Verteilung der Fehler dar, die bei der Bearbeitung der Aufgaben im Interaction Room gemacht wurden. Die dritte und vierte Box visualisieren die Fehler,

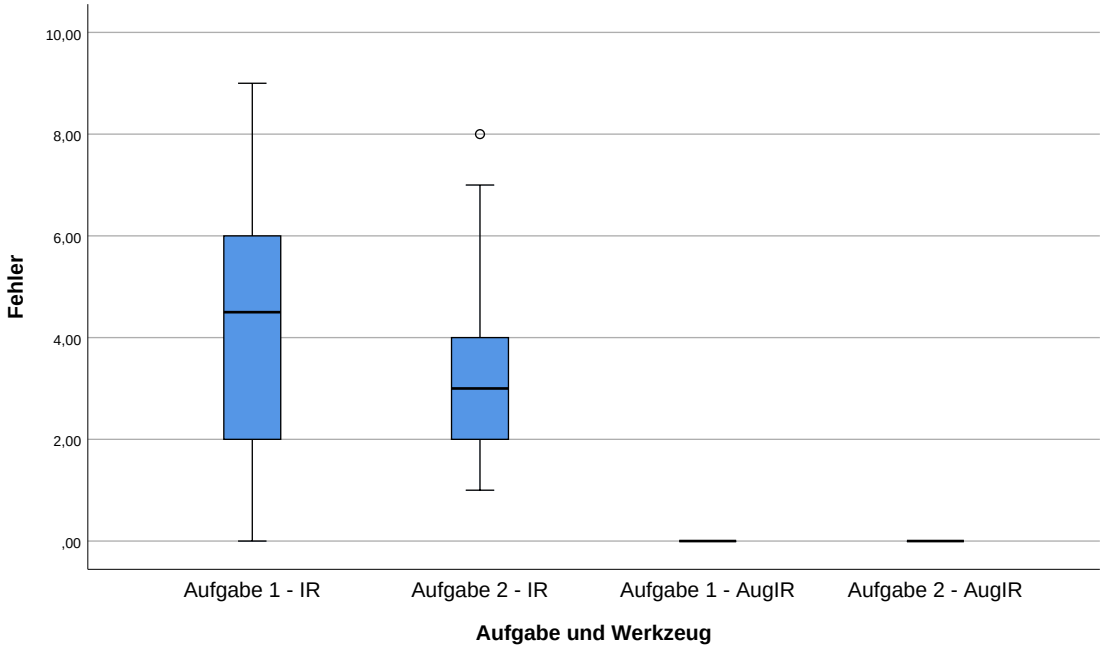

Abbildung 8.2.: Boxplot für die Verteilung der Fehler, die bei der Bearbeitung der Aufgaben in beiden Gruppen gemacht wurden.

die bei der Bearbeitung der Aufgaben im AugIR gemacht wurden. Es wird noch einmal deutlich, dass nur ein einziger Teilnehmer als Ausreißer die erste Aufgabe im Interaction Room fehlerfrei lösen konnte. Bei der zweiten Aufgabe gelang dies niemanden. Demgegenüber konnten im AugIR alle Aufgaben von den Teilnehmern vollständig ohne Fehler abgeschlossen werden.

Tabelle 8.4 fasst die wesentlichen deskriptiven Statistiken für beide Werkzeuge als Übersicht zusammen. Die ersten beiden Zeilen geben das Minimum und Maximum für die jeweils zur Bearbeitung der entsprechenden Aufgabe benötigte Zeit in Sekunden und die Anzahl der dabei gemachten Fehler an. Die dritte Zeile zeigt den entsprechenden Mittelwert $\overline{x}$ und in der vierten Zeile ist die Standardabweichung s aufgeführt. Die Formeln zur Berechnung der jeweiligen Kennzahlen finden sich in Abschnitt 3.2.9.

Man sieht sofort, dass die Teilnehmer zur Bearbeitung der Aufgaben im AugIR durchschnittlich wesentlich weniger Zeit benötigten als im Interaction Room. Wäh-

Tabelle 8.4.: Deskriptive Statistiken für die Bearbeitung der Aufgaben im Interaction Room und AugIR.

	Interaction Room				AugIR			
	Aufgabe 1		Aufgabe 2		Aufgabe 1		Aufgabe 2	
	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler	Zeit	Fehler
Min	720	0	475	1	225	0	95	0
Max	1 814	9	1 564	8	685	0	231	0
$\overline{x}$	1 309,05	4,09	949	3,23	437,09	0	149,18	0
s	282,2	2,37	235,79	1,85	125,44	0	34,68	0

rend der schnellste Teilnehmer die erste Aufgabe im Interaction Room in 12 Minuten bearbeiten konnte, benötigte selbst der langsamste Teilnehmer im AugIR hierzu lediglich 11 Minuten und 25 Sekunden. Der langsamste Teilnehmer im AugIR konnte die erste Aufgabe somit um 35 Sekunden schneller lösen als der schnellste Teilnehmer im Interaction Room. Für die beiden schnellsten Teilnehmer ergibt sich sogar eine zeitliche Differenz von 8 Minuten und 15 Sekunden.

Dieser Unterschied wird auch für die zweite Aufgabe deutlich: Diese konnte im Interaction Room vom schnellsten Teilnehmer in 7 Minuten und 55 Sekunden gelöst werden. Der langsamste Teilnehmer im AugIR benötigte hierzu jedoch lediglich 3 Minuten und 51 Sekunden. Somit war der langsamste Teilnehmer im AugIR 4 Minuten und 4 Sekunden schneller als der schnellste Teilnehmer im Interaction Room. Für die beiden schnellsten Teilnehmer ergibt sich eine zeitliche Differenz von 6 Minuten und 20 Sekunden.

Während die Teilnehmer im Interaction Room bei der Bearbeitung der ersten Aufgabe durchschnittlich etwa 4 Fehler und bei der Bearbeitung der zweiten Aufgabe durchschnittlich etwa 3 Fehler machten, konnten alle Teilnehmer beide Aufgaben im AugIR vollständig fehlerfrei lösen.

Die vorangegangene Präsentation der Ergebnisse zeigt deutlich, dass der AugIR die Stakeholder beim Auffinden von zusammenhängenden Elementen in den Skizzen unterstützt hat und ihnen dabei helfen konnte, Fehler zu vermeiden. Alle Teilnehmer waren bei der Bearbeitung der Aufgaben im AugIR schneller als im Interaction

Room. Dabei konnten sie die Aufgaben im AugIR vollständig fehlerfrei lösen, was im Interaction Room nicht möglich war. Der Form und Vollständigkeit halber findet sich ergänzend hierzu in den Abschnitten A.3 und A.4 im Anhang eine formale Analyse der Ergebnisse mit Mitteln der deskriptiven Statistik.

8.6. Auswertung der Fragebögen

Im Anschluss an die Studie wurde jeder Teilnehmer gebeten, einen Fragebogen auszufüllen, um die Unterstützung bei der Erkennung von inhaltlichen Zusammenhängen zu bewerten. Jeder Fragebogen enthielt 16 zu bewertende Aussagen sowie ein Freitextfeld, in dem die Teilnehmer beschreiben konnten, was ihnen am AugIR besonders gut oder gar nicht gefiel.

Um die Teilnehmer beim Ausfüllen des Fragebogens nicht durch die Formulierung der Fragestellungen zu beeinflussen, wurden diese bewusst neutral gewählt. Anstatt einer wertenden Aussage zuzustimmen oder diese abzulehnen, konnte eine objektive aber unvollständige Aussage auf einer 5-stufigen Likert-Skala [144] durch eine passende Wertung komplettiert werden.

Hierbei wurde bewusst eine ungerade Anzahl von Antwortmöglichkeiten vorgegeben, damit unentschlossene Teilnehmer auch eine neutrale Antwort wählen konnten. Weil gezeigt werden soll, dass die digitalen Zeichenflächen im AugIR mindestens so gut funktionieren wie analoge Whiteboards, ist es in vielen Fällen ausreichend, wenn beide Werkzeuge als gleich gut bewertet werden.

Das Ausfüllen der Fragebögen geschah wie eingangs erläutert vollständig anonym, um möglichst objektive Antworten der Teilnehmer zu erhalten. Hierzu wurden die Fragebögen im Anschluss an jeden Versuchsdurchlauf ohne Aufsicht ausgefüllt und in einer verschlossenen Box gesammelt, die erst nach Beendigung des Gesamtexperiments geöffnet wurde.

Die Ergebnisse der Befragung sind in Tabelle 8.5 dargestellt.

Tabelle 8.5.: Fragebogen zur Bewertung der Unterstützung bei der Erkennung von inhaltlichen Zusammenhängen im AugIR.

#	Aussage	Bewertung				
		++	+	o	-	--
1	Das Erkennen von Zusammenhängen auf analogen Whiteboards war...	Leicht				Schwierig
		1	7	3	8	3
2	Das Erkennen von Zusammenhängen auf analogen Whiteboards war...	Schnell			Zeitlich aufwändig	
		0	0	4	6	12
3	Das Erkennen von Zusammenhängen im AugIR war...	Leicht				Schwierig
		20	2	0	0	0
4	Das Erkennen von Zusammenhängen im AugIR war...	Schnell			Zeitlich aufwändig	
		21	1	0	0	0
5	Für das Erkennen von Zusammenhängen im AugIR war die Unterstreichung von Elementen...	Hilfreich			Nicht hilfreich	
		18	4	0	0	0
6	Für das Erkennen von Zusammenhängen im AugIR war die Vorschauansicht auf verwandte Elemente...	Hilfreich			Nicht hilfreich	
		22	0	0	0	0

Tabelle 8.5.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
7	Für das Erkennen von Zusammenhängen im AugIR war die Navigation zwischen verwandten Elementen...	Hilfreich				Nicht hilfreich
		18	3	0	1	0
8	Die Navigation zwischen verwandten Elementen im AugIR war...	Schnell				Zeitlich aufwändig
		20	2	0	0	0
9	Die Navigation zwischen verwandten Elementen im AugIR war...	Intuitiv				Nicht intuitiv
		17	5	0	0	0
10	Insgesamt empfinde ich die Bedienung des AugIRs als...	Intuitiv				Nicht intuitiv
		16	6	0	0	0
11	Den Umgang mit dem AugIR zu erlernen war...	Leicht				Schwierig
		17	5	0	0	0
12	Insgesamt halte ich den AugIR für...	Nützlich				Nicht nützlich
		20	2	0	0	0
13	Verglichen mit analogen Whiteboards war das Erkennen von Zusammenhängen im AugIR...	Leichter				Schwieriger
		19	2	1	0	0

Tabelle 8.5.: (Fortsetzung)

#	Aussage	Bewertung				
		++	+	o	-	--
14	Verglichen mit analogen Whiteboards war das Erkennen von Zusammenhängen im AugIR...	Schneller				Zeitlich aufwändiger
		21	1	0	0	0
15	Verglichen mit analogen Whiteboards hat der AugIR mich beim Erkennen von Zusammenhängen...	Unterstützt				Behindert
		21	1	0	0	0
16	Insgesamt würde ich folgendes Werkzeug präferieren...	AugIR				Analoge Whiteboards
		15	6	1	0	0

Aus der Tabelle geht hervor, dass viele Teilnehmer das Erkennen von Zusammenhängen auf analogen Whiteboards als schwierig und zeitlich aufwändig empfanden, während dies im AugIR von fast allen als sehr leicht und sehr schnell wahrgenommen wurde. Als besonders hilfreich wurden hierbei die Visualisierung der Trace Links durch Unterstreichung, die Vorschauansicht auf verwandte Elemente sowie die Navigation zwischen diesen bewertet.

Alle Teilnehmer empfanden die Bedienung des AugIRs als intuitiv oder sehr intuitiv. Dementsprechend war die überwiegende Mehrheit der Teilnehmer der Ansicht, dass der Umgang mit dem AugIR sehr leicht zu erlernen sei.

Insgesamt bewerteten fast alle Teilnehmer den AugIR als sehr nützlich und würden ihn gegenüber traditionellen analogen Whiteboards bevorzugen.

Zusätzlich zur Bewertung der vorgegebenen Aussagen konnten die Teilnehmer in einem Freitextfeld beschreiben, was ihnen am AugIR besonders gut oder gar nicht

gefiel. Die wesentlichen Antworten, die teilweise auch von mehreren Teilnehmern unabhängig voneinander gegebenen wurden, werden im Folgenden kurz zusammengefasst:

Viele Teilnehmer lobten die leichte und intuitive Bedienbarkeit der AugIR-Software. Insbesondere das schlichte und bewusst minimalistisch gehaltene User Interface fand dabei großen Zuspruch, weil es anders als in vielen klassischen CASE-Tools nicht überladen wirke und Benutzer deshalb nicht von ihrer eigentlichen Aufgabe ablenke. Viele Aktionen wie beispielsweise das Zoomen und Verschieben der Zeichenfläche seien dabei über intuitive Gesten ausführbar, die von anderen berührungssensitiven Geräten bekannt seien.

Auch die Visualisierung der Trace Links durch Unterstreichung der zusammengehörigen Elemente wurde von mehreren Teilnehmern explizit gelobt. Die verwendete Hyperlink-Metapher sei intuitiv verständlich und die Teilnehmer hoben insbesondere positiv hervor, dass hierbei jeweils nur die Verknüpfungen angezeigt würden, die tatsächlich für den aktuell betrachteten Ausschnitt einer Skizze relevant seien. Durch zentrierte Darstellung der verknüpften Elemente in der Vorschauansicht sei dabei sofort und intuitiv ersichtlich, zu welchen Inhalten ein Bezug existiert.

Als negativ wurde von den Teilnehmern hingegen bewertet, dass die Vorschauansicht beim Antippen eines Trace Links nicht immer an der erwarteten Stelle erschien. Zur Ermittlung einer geeigneten Position wurde ein spezieller Algorithmus implementiert, der die bestmögliche Position anhand mehrere Faktoren bestimmt. Hierdurch sollte die Vorschauansicht möglichst wenige potentiell relevante Inhalte auf dem elektronischen Whiteboard überdecken. Leider ist dies nicht immer problemlos möglich: Wenn beispielsweise ein Trace Link am Rand des Bildschirms angezeigt und ausgewählt wird, so muss die zugehörige Vorschau teilweise in einiger Entfernung zum betreffenden Element platziert werden, um dieses nicht zu überdecken und nicht über den Bildschirmrand hinaus zu ragen.

Weiterhin wurde von vielen Teilnehmern die verwendete Hardware kritisiert. Weil es sich um ein Universitätslabor handelt, entspricht diese in vielen Punkten nicht dem aktuellen Stand der Technik und ist teilweise veraltet. Dies wurde insbesondere bei der Auflösung der Beamer deutlich: Diese ist technisch bedingt sehr gering,

wodurch einige Inhalte schlecht lesbar sind. Weil es sich bei der AugIR-Software um einen Prototyp handelt, der nicht explizit auf Geschwindigkeit optimiert ist, reagierte zudem die Software teilweise leicht verzögert auf Eingaben der Benutzer, was einigen Teilnehmern negativ auffiel.

8.7. Gefahren für die Gültigkeit

Wie in Abschnitt 8.2 beschrieben wurden die beiden betrachteten Projekte bewusst so gewählt, dass sie möglichst repräsentativ und leicht verständlich sind. Dennoch enthalten auch sie einige Fachbegriffe und Abkürzungen, die möglicherweise nicht jedem Teilnehmer vor Beginn der Studie geläufig waren. In einem Projekt erscheint beispielsweise die Abkürzung „ADM“. Aus dem Kontext geht schnell hervor, dass dies für „Außendienstmitarbeiter“ steht. Abhängig von Vorwissen und Allgemeinbildung könnten solche Begriffe einigen Teilnehmern aber dennoch das Verständnis der skizzierten Inhalte geringfügig erschwert haben.

Darüber hinaus ist ein geschlechtsbezogener Verzerrungseffekt (engl. Gender Bias) nicht auszuschließen, weil nur drei weibliche aber 19 männliche Probanden an der Studie teilnahmen und das Ergebnis maßgeblich vom individuellen inhaltlichen Textverständnis abhing. Die Hintergründe der Teilnehmer waren aber sehr verschieden und die Diskussion der Resultate in Abschnitt 8.5 machte deutlich, dass zumindest etwaige Vorkenntnisse der Interaction-Room-Methode das Ergebnis nicht positiv beeinflussten.

Es kann jedoch nicht gänzlich ausgeschlossen werden, dass die Antworten einiger Teilnehmer im Fragebogen besonders positiv ausfielen, weil sie den Verantwortlichen hiermit einen Gefallen tun wollten. Um diese Gefahr so weit wie möglich zu minimieren, wurden die Fragebögen wie eingangs erläutert vollständig anonym ohne Aufsicht ausgefüllt und in einer verschlossenen Box gesammelt, die erst nach Beendigung des Gesamtexperiments geöffnet wurde.

Während die Teilnehmer im AugIR an einem einzigen elektronischen Whiteboard arbeiten und dort die Skizzen interaktiv durchschalten konnten, mussten die analogen

Skizzen aus Platzgründen auf mehrere Whiteboards verteilt werden. Im Verlauf der Studie mussten die Teilnehmer sich deshalb im Interaction Room ständig zwischen den Whiteboards hin und her bewegen, während sie im AugIR starr an einer Stelle stehen bleiben konnten. Studien haben allerdings gezeigt, dass körperliche Bewegung bei der Arbeit mit großen Informationsmengen einen positiven Effekt haben kann [16, 17]. Die Bewegung, die mit der Arbeit an den analogen Skizzen im Interaction Room einherging, könnte somit das Ergebnis der Aufgaben ebenfalls geringfügig beeinflusst haben.

Insgesamt kann aufgrund der Repräsentativität der gewählten Projekte und der Heterogenität der Teilnehmergruppen aber angenommen werden, dass die Resultate der Studie auf andere Interaction-Room-Projekte und Stakeholder übertragbar sind. Die Verwendung von Trace Links sollte daher in anderen Projekten ähnlich gut funktionieren und auch dort bei der Identifikation von inhaltlichen Zusammenhängen unterstützen können.

8.8. Zusammenfassung und Diskussion

In diesem Kapitel wurden die Resultate einer Studie vorgestellt, welche die Verwendung von Trace Links in Interaction-Room-Projekten betrachtet. Hierzu wurden zwei Teilnehmergruppen gebeten, inhaltliche Zusammenhänge zwischen Skizzen aufzudecken. Während eine Gruppe hierbei traditionelle analoge Whiteboards verwenden und ohne technische Hilfsmittel auskommen musste, konnte die andere Gruppe auf elektronische Whiteboards zurückgreifen und wurde von der AugIR-Software unterstützt.

Beide Gruppen mussten jeweils zwei Aufgaben bearbeiten: In der ersten Aufgabe waren textuelle Zusammenhänge zwischen Skizzen zu identifizieren, während in der zweiten Aufgabe mehrfach vorkommende Annotationsnummern gefunden werden mussten.

Es konnte gezeigt werden, dass inhaltliche Zusammenhänge zwischen Skizzen im AugIR wesentlich schneller erkannt werden können als auf analogen Whiteboards.

Hierbei war die Identifikation von textuellen Zusammenhängen in Aufgabe 1 durchschnittlich aufwändiger und schwieriger als die Identifikation von gleichen Annotationsnummern in Aufgabe 2. Dies ist plausibel, wenn man bedenkt, dass für die Lösung von Aufgabe 1 ein Verständnis der vorliegenden Texte notwendig war, während für die Lösung von Aufgabe 2 eine simple Suche nach identischen Zahlen ausreichte.

Weiterhin wurde deutlich, dass beim Auffinden von Zusammenhängen auf analogen Whiteboards durchschnittlich wesentlich mehr Fehler passieren als im AugIR. Während alle Teilnehmer beide Aufgaben im AugIR vollständig fehlerfrei lösen konnten, gelang dies im Interaction Room niemandem.

Hierbei ist insbesondere hervorzuheben, dass keiner der beiden erfahrenen Interaction-Room-Coaches, die am Experiment teilnahmen, die Aufgaben bei der Verwendung analoger Whiteboards fehlerfrei lösen konnte. Beide machten hierbei jeweils etwa durchschnittlich viele Fehler. Dies verdeutlicht, dass auch ein geschulter Experte mit jahrelanger Erfahrung leicht inhaltliche Zusammenhänge zwischen analogen Skizzen übersehen kann.

Durch eine ausführliche Analyse mit Mitteln der deskriptiven Statistik (vgl. Abschnitte A.3 und A.4 im Anhang) konnte gezeigt werden, dass die ermittelten Ergebnisse statistisch signifikant sind. Insgesamt bestätigt dies die eingangs aufgestellten Hypothesen. Es lässt sich deshalb abschließend festhalten, dass der AugIR die Stakeholder bei der schnellen und insbesondere fehlerfreien Identifikation von inhaltlichen Zusammenhängen zwischen Skizzen unterstützt.

Teil IV.

Schlussteil

9. Fazit

In diesem Kapitel werden zunächst die Beiträge der vorliegenden Arbeit kurz zusammengefasst. Danach wird diskutiert, inwiefern dies die in Abschnitt 1.3 identifizierten Probleme bei der Arbeit mit Freihandskizzen löst und die in Abschnitt 1.5 formulierte Forschungshypothese bestätigt.

9.1. Zusammenfassung der Beiträge

In dieser Arbeit wurde das Konzept des Augmentierten Interaction Rooms (AugIR) präsentiert. Hierbei handelt es sich um einen technisch augmentierten Team-Raum, der verschiedene Probleme bei der Arbeit mit analogen Freihandskizzen lösen kann.

Diese Probleme, die bei der Durchführung und Nachbereitung von Workshops auftreten, wurden zunächst in Abschnitt 1.3 analysiert. Hierbei konnten zwölf wesentliche Einschränkungen und Nachteile identifiziert werden, die sich aus der Verwendung rein analoger Zeichenwerkzeuge ergeben. Diese wurden jeweils am Beispiel der Interaction-Room-Methode illustriert.

In Kapitel 3 wurde daraufhin ein Konzept zur Lösung dieser Probleme entwickelt, welches auf der Verwendung elektronischer Whiteboards in Kombination mit mobilen Endgeräten basiert. Damit Stakeholder dieses als adäquaten Ersatz für traditionelle Whiteboards akzeptieren, ist jedoch die Konzeption spezieller Lösungen für mehrere nicht-triviale Probleme erforderlich:

Um gezeichnete Inhalte und handschriftlich geschriebene Texte erkennen und interpretieren zu können, wird zunächst ein Algorithmus benötigt, der zuverlässig

zwischen Texten und Formen unterscheiden kann. Ein entsprechender Algorithmus zur Klassifizierung handgeschriebener Inhalte wurde in Abschnitt 3.4 präsentiert. Hierzu wurden 24 charakteristische lokale und globale Eigenschaften ermittelt, aus denen ein numerischer Vektor $f(L_i) := (f_1(L_i), \ldots, f_{24}(L_i)) \in \mathbb{R}^{24}$ für jeden Linienzug L_i bestimmt werden kann. Mit Hilfe der logistischen Regression kann ein geeignetes Machine-Learning-System trainiert werden, das in der Lage ist, den Typ eines Linienzugs L_i anhand seines beschreibenden Vektors $f(L_i)$ zu bestimmen. Eine empirische Bewertung der Features in Abschnitt 3.4.6 hat gezeigt, dass für die betrachtete Datenbasis im Durchschnitt 99,65 % aller Text-Linienzüge und 93,24 % aller Form-Linienzüge korrekt klassifiziert werden können.

Darüber hinaus müssen alle gezeichneten Linienzüge nach logischer Zusammengehörigkeit gruppiert werden. Dies ist eine wesentliche Voraussetzung für eine funktionierende Handschrifterkennung und ermöglicht zudem eine sinnvolle Interaktion mit den gezeichneten Inhalten, indem zusammengehörende Linienzüge beispielsweise als einzelnes Element modifiziert oder auf der Zeichenfläche verschoben werden können. Hierzu wurde in Abschnitt 3.5 ein Algorithmus entwickelt, der Linienzüge automatisch anhand ihrer räumlichen und zeitlichen Nähe gruppiert. In Anlehnung an verwandte Forschungsarbeiten wurde hierbei eine maximale zeitliche Differenz von 3,5 Sekunden und ein maximaler räumlicher Abstand von 10 Pixeln zwischen benachbarten Linienzügen gewählt [59, 238].

Damit auf den klassifizierten und gruppierten Text-Linienzügen eine Handschrifterkennung durchgeführt werden kann, müssen diese darüber hinaus in geeigneter Weise in Tupel aufgetrennt werden, die jeweils ein oder mehrere Elemente umfassen. Dies ist ein wichtiger Schritt, der in der digitalen Sprachverarbeitung als Tokenisierung bezeichnet wird [233]. In Abschnitt 3.6 wurden deshalb drei geeignete heuristische Regeln t_1, t_2 und t_3 zur Tokenisierung von Texten entwickelt, die zudem gängigen Problemen bei der Handschrifterkennung entgegenwirken. Die Bewertung der aufgestellten Regeln durch ein Experiment, welches in Abschnitt 7.3.2 präsentiert wurde, machte deutlich, dass die Erfassung textueller Beziehungen zwischen skizzierten Inhalten durch Verwendung dieser Regeln signifikant verbessert werden kann.

Weil sich jedoch nicht alle potentiellen Schwierigkeiten bei der Handschrifterken-

nung durch den Einsatz geeigneter Tokenisierungsregeln lösen lassen, ist es sinnvoll, die erzeugten Tokenisierungen darüber hinaus in einem anschließenden Schritt zu modifizieren. Der entsprechende Algorithmus wurde in Abschnitt 3.7 präsentiert. Die Grundidee besteht darin, Wörter einer Tokenisierung, die in anderen Skizzen nicht vorkommen, durch hinreichend ähnliche Wörter aus diesen Skizzen zu ersetzen. Auf diese Weise kann die Korrelation zwischen potentiell zusammenhängenden Elementen unterschiedlicher Skizzen erhöht werden, wenn die entsprechenden Texte Handschrifterkennungsfehler enthalten oder einzelne Wörter eine unterschiedliche Schreibweise aufweisen. In Abschnitt 3.7.2 konnte im Rahmen eines Experiments gezeigt werden, dass für die betrachtete Datenbasis bei Verwendung eines Ähnlichkeitsschwellwerts von $\mu = 0{,}75$ hierdurch insgesamt 57,14 % aller falsch erkannten Wörter korrigiert werden können.

Um Stakeholder bei der effizienten Navigation zwischen inhaltlich zusammenhängenden Elementen sowie bei der Identifikation von potentiellen Widersprüchen, Inkonsistenzen und Projektrisiken zu unterstützen, wurde in Abschnitt 3.8 ein Algorithmus zur Erfassung von Trace Links zwischen Freihandskizzen entwickelt. Dieser verwendet die zuvor erzeugten und modifizierten Tokenisierungen und bestimmt mit Hilfe der Kosinus-Ähnlichkeit die textuelle Ähnlichkeit zwischen Skizzenelementen.

Basierend auf den bisherigen Erkenntnissen wurde in Abschnitt 3.9 beschrieben, wie Freihandskizzen im AugIR erstellt, bearbeitet und annotiert werden können. Darüber hinaus wurden im Hinblick auf die in Abschnitt 1.3 geschilderten Probleme und eine vollständige Digitalisierung der Interaction-Room-Methode zwei zusätzliche digitale Ansichten konzipiert:

In Abschnitt 3.9.1 wurde erläutert, wie durch Anbindung mobiler Endgeräte ein digitaler Feature Canvas entstehen kann. Stakeholder können hierbei Feature-Karten auf individuellen mobilen Endgeräten schreiben und drahtlos auf die elektronischen Whiteboards übertragen. Auf diese Weise können analoge Karteikarten und Metaplanwände sinnvoll im AugIR abgebildet werden.

Weiterhin wurde in Abschnitt 3.9.2 ein Konzept für ein digitales Storyboard vorgestellt. Dieses erlaubt nicht nur die Umsetzung des in der Interaction-Room-Me-

thode spezifizierten Interaction Canvas, sondern ermöglicht zudem die Definition komplexer benutzerdefinierter Gesten zur Definition von Dialogübergängen sowie eine interaktive Ausführung dieser in einem speziellen Simulationsmodus.

Zur effizienten Verwaltung der skizzierten Inhalte wurde in Abschnitt 3.10 eine als Navigationsgraph bezeichnete Datenstruktur eingeführt. Diese speichert alle Skizzen und deren Beziehungen zueinander und ermöglicht eine übersichtliche hierarchische Darstellung der Inhalte. Sie bietet zahlreiche Vorteile gegenüber verwandten Ansätzen wie beispielsweise eines dynamischen Filmstreifens [219, 220] oder eines festen zweidimensionalen Gitters [156], die ebenfalls ausführlich in Abschnitt 3.10 diskutiert wurden.

In Abschnitt 3.11 wurde erläutert, wie Verfeinerungen und Trace Links visualisiert und zur vertikalen und horizontalen Navigation zwischen Skizzen verwendet werden können. Die vertikale Navigation ist durch dynamische Overlays realisiert, die es nicht nur ermöglichen, schnell zwischen verschiedenen Abstraktionsniveaus zu wechseln, sondern darüber hinaus auch bis zu drei Abstraktionsebenen auf einem Whiteboard gleichzeitig darstellen zu können. Trace Links zur horizontalen Navigation werden durch eine Hyperlink-Metapher visualisiert, indem verknüpfte Elemente farblich unterstrichen werden. Dies hilft Stakeholdern bei der Identifikation von inhaltlichen Zusammenhängen, die ansonsten möglicherweise unentdeckt geblieben wären und unterstützt sie darüber hinaus bei der schnellen Navigation zwischen verbundenen Inhalten.

Weiterhin kommt im AugIR eine umfangreiche Impact Analysis zum Einsatz, die im Detail in Abschnitt 3.12 beschrieben wurde. Diese unterstützt Stakeholder bei der Sicherstellung der Konsistenz hinsichtlich der platzierten Annotationen sowie bei der Identifikation potentieller Widersprüche und Projektrisiken. Hierzu werden Annotationen als sogenannte indirekte Annotationen auf verbundene Elemente übertragen, was insbesondere bei der Aufdeckung potentieller Widersprüche hilft, die sich aus Annotationen ergeben, die in unterschiedlichen aber inhaltlich zusammenhängenden Skizzen platziert wurden. Darüber hinaus können Annotationen über Trace Links zu ausgewählten Elementen aggregiert werden, um auf diese Weise bei der Identifikation von Projektrisiken zu unterstützen, die sich aus den platzierten Annotationen

ableiten lassen.

Um eine Evaluation des in Kapitel 3 entwickelten AugIR-Konzepts zu ermöglichen, wurde dieses prototypisch umgesetzt. Technische Details zur Implementierung wurden in Kapitel 4 präsentiert.

Eine umfangreiche Evaluation des entwickelten Konzepts erfolgte daraufhin in Teil III dieser Arbeit.

In Kapitel 5 wurde zunächst die Evaluation der grundlegenden Funktionalitäten zur Bearbeitung und Annotierung von Skizzen beschrieben. Es konnte gezeigt werden, dass dies im AugIR sowohl schnell als auch intuitiv möglich ist. Die Probanden der Studie erachteten den AugIR als nützliches und intuitives Werkzeug und akzeptierten ihn als Ersatz für traditionelle analoge Whiteboards.

In Kapitel 6 wurde eine weitere Nutzerstudie zur Evaluation des Interaction Canvas vorgestellt. Diese machte deutlich, dass der AugIR von den Teilnehmern der Studie als hilfreiches Werkzeug zur Erstellung digitaler Storyboards angesehen wurde. Gesten waren im AugIR wesentlich intuitiver und schneller verständlich als auf analogen Whiteboards und eine abschließende Umfrage hat ergeben, dass die meisten Stakeholder den AugIR deshalb zur Erstellung von Storyboards traditionellen Whiteboards vorziehen würden.

Kapitel 7 untersuchte die erreichbare Qualität bei der Trace-Link-Erfassung. Es wurde gezeigt, dass eine kombinierte Anwendung der Tokenisierungsregeln t_1, t_2 und t_3 auf die erkannten Handschrifttexte vor Erfassung der Trace Links die Qualität des Algorithmus signifikant verbessern kann. Bei Verwendung eines Kosinus-Ähnlichkeitsschwellwerts von $\nu := 0{,}55$ konnte durchschnittlich eine Precision von 93 % und ein Recall von 86 % erreicht werden. Für jedes Projekt der analysierten Datenbasis werden somit im Durchschnitt 43,88 Trace Links erfasst, von denen 40,69 korrekt sind. Pro Projekt existieren durchschnittlich lediglich 3,19 False Positives und 4,5 übersehene Trace Links. Der Algorithmus war somit in der Lage, die meisten relevanten Trace Links in fast jedem untersuchten Projekt aufzudecken und dabei nur eine geringe Anzahl falsch positiver Treffer zu produzieren.

Tabelle 9.1.: Zwölf Nachteile und Einschränkungen, die sich aus der Verwendung rein analoger Zeichenwerkzeuge ergeben.

Keine Unterstützung für Modifikation der Inhalte
Beschränkte Größe und Anzahl der Zeichenflächen
Existenz von Medienbrüchen
Unzureichende Verhaltensbeschreibung von Benutzerschnittstellen
Keine Unterstützung für Verwaltung der skizzierten Inhalte
Keine Unterstützung für vertikale Navigation
Keine Unterstützung für horizontale Navigation
Keine Unterstützung für Entwurf alternativer Lösungen
Aufwändige Persistierung der Inhalte
Keine Sicherstellung der Konsistenz
Keine Unterstützung für Identifikation von Widersprüchen
Keine Unterstützung für Identifikation von Projektrisiken

Abschließend wurde in Kapitel 8 die tatsächliche Nützlichkeit der Trace Links im praktischen Einsatz bestätigt. Im Rahmen einer umfangreichen Studie konnte gezeigt werden, dass inhaltliche Zusammenhänge zwischen Skizzen im AugIR wesentlich schneller erkannt werden konnten als auf analogen Whiteboards. Darüber hinaus machten Stakeholder hierbei im AugIR wesentlich weniger Fehler. Durch eine ausführliche Analyse mit Mitteln der deskriptiven Statistik ließ sich überdies nachweisen, dass die ermittelten Ergebnisse statistisch signifikant waren.

9.2. Diskussion

Im Folgenden soll kurz diskutiert werden, wie der AugIR die in Abschnitt 1.3 beschriebenen Probleme löst und wie dies die in Abschnitt 1.5 formulierte Forschungshypothese bestätigt.

Tabelle 9.1 listet zunächst noch einmal zusammengefasst alle identifizierten Nachteile und Einschränkungen auf, die sich aus der Verwendung rein analoger Zeichenwerkzeuge ergeben. Diese werden durch den AugIR wie folgt behoben:

Unterstützung für Modifikation der Inhalte Die digitalen Whiteboards im AugIR bieten eine umfassende Unterstützung für die Modifikation der gezeichneten Inhalte. Elemente können selektiert und frei auf der Zeichenfläche verschoben werden, ohne dass dies Auswirkungen auf andere Teile der editierten Skizze hat. Dabei können alle Änderungen jederzeit durch ein umfangreiches Undo-/Redo-System rückgängig gemacht und wiederhergestellt werden. Dies wurde in Abschnitt 3.9 beschrieben.

Unbeschränkte Größe und Anzahl der Zeichenflächen Die Größe der Zeichenflächen ist theoretisch unbeschränkt und sie können stufenlos gezoomt und frei in alle Richtungen verschoben werden. In jedem Projekt können dabei beliebig viele Skizzen erstellt werden, wobei der Navigationsgraph den Stakeholdern hilft, diese übersichtlich zu strukturieren und zu verwalten. Dies wurde in den Abschnitten 3.9 und 3.10 beschrieben.

Vermeidung von Medienbrüchen Medienbrüche werden vom AugIR vollständig eliminiert, indem alle Inhalte direkt in digitaler Form erfasst werden. Skizzen können direkt auf den elektronischen Whiteboards unter Zuhilfenahme spezieller Stifte gezeichnet und bearbeitet werden. Feature-Karten, die im Interaction Room ansonsten auf analoge Karteikarten geschrieben und an Metaplanwände geheftet wurden, lassen sich auf mobilen Endgeräten verfassen und drahtlos an die digitalen Whiteboards übertragen. Dies wurde in Abschnitt 3.9.1 beschrieben.

Verhaltensbeschreibung von Benutzerschnittstellen Der AugIR stellt ein digitales Storyboard zur Skizzierung von grafischen Benutzerschnittstellen zur Verfügung. Durch die Definition benutzerdefinierter Gesten können darauf insbesondere Transitionen zwischen Skizzen spezifiziert werden, die durch die entsprechenden Gesten ausgelöst werden. Beides kann interaktiv in einem speziellen Simulationsmodus getestet werden, um auf diese Weise ein besseres Gefühl für das Look and Feel einer Anwendung und ein adäquates Verständnis der Dialogflüsse zu erhalten. Zusätzliche digitale Werkzeuge zur Erstellung von Mock-ups und Klick-Prototypen werden hierdurch obsolet. Dies wurde in Abschnitt 3.9.2 beschrieben.

Verwaltung von skizzierten Inhalten Die skizzierten Inhalte werden im Navigationsgraph gespeichert und übersichtlich verwaltet. Skizzen können zu logischen Gruppen zusammengefasst werden, die sich bei Bedarf ein- und ausklappen lassen, um den Detailgrad der dargestellten Inhalte an die aktuellen Bedürfnisse der Stakeholder anzupassen. Darüber hinaus visualisiert der Navigationsgraph alle Skizzen in Form von Vorschaubildern, die frei auf dem elektronischen Whiteboard angeordnet werden können. Dies ermöglicht Stakeholdern eine gleichzeitige Betrachtung von mikroskopischen und makroskopischen Aspekten eines Softwareprojekts sowie eine zielgerichtete Navigation zu bestimmten Skizzen. Dies wurde in Abschnitt 3.10 beschrieben.

Unterstützung der vertikalen Navigation Zur Unterstützung der vertikalen Navigation und zur Modellierung unterschiedlicher Abstraktionsebenen können Skizzenknoten im Navigationsgraph durch eine Verfeinerung miteinander verknüpft werden. Dies wird in den entsprechenden Skizzen durch ein passendes Verfeinerungssymbol dargestellt, über welches Stakeholder schnell zwischen den verbundenen Inhalten navigieren können. Überdies können Verfeinerungen und Vergröberungen in dynamischen Overlays geöffnet werden, die über der aktuell angezeigten Skizze schweben. Dies erlaubt die gleichzeitige Darstellung von bis zu drei Abstraktionsebenen auf dem gleichen Whiteboard. Dies wurde in Abschnitt 3.11.1 beschrieben.

Unterstützung der horizontalen Navigation Durch Trace Links, die in Form von Hyperlinks visualisiert werden, unterstützt der AugIR außerdem die horizontale Navigation zwischen zusammenhängenden Inhalten. Zusammengehörige Elemente werden farblich unterstrichen und durch Antippen können Stakeholder eine Vorschau auf alle verknüpften Elemente erhalten. Auf diese Weise können verbundene Inhalte inspiziert werden, ohne dass diese zunächst umständlich gesucht oder explizit zur Bearbeitung geöffnet werden müssen. Durch Auswahl einer Vorschau können Stakeholder überdies schnell und intuitiv zur entsprechenden Skizze navigieren. Dies wurde in Abschnitt 3.11.2 beschrieben.

Unterstützung für Entwurf alternativer Lösungen Der AugIR unterstützt den Entwurf alternativer Lösungsideen und experimenteller Skizzen, indem alle Inhalte

auf Knopfdruck vervielfältigt werden können. Stakeholder können somit von jeder Skizze leicht beliebig viele Kopien anfertigen, diese individuell weiterentwickeln und bei Bedarf übernehmen oder verwerfen. Auf jedem Whiteboard können hierzu beliebig viele Skizzen geöffnet und zum Vergleich nebeneinander angezeigt werden. Änderungen lassen sich hierbei jederzeit durch das implementierte Undo-/Redo-System rückgängig machen und wiederherstellen, weshalb Stakeholder nicht zögerlich bei der spontanen Skizzierung neuer Ideen sein müssen. Dies wurde in Abschnitt 3.9 beschrieben.

Automatische Persistierung der Inhalte Alle Skizzen werden in regelmäßigen Abständen automatisch gespeichert, so dass sich Stakeholder vollständig auf den Modellierungsprozess fokussieren können, ohne dabei an die Persistierung der Inhalte denken zu müssen. Die erstellten Skizzen können auf Knopfdruck als Vektorgrafiken exportiert werden, so dass sie leicht in entsprechenden Grafikprogrammen nachbearbeitet oder direkt in abschließenden Dokumentationen eingebunden werden können. Dies wurde ebenfalls in Abschnitt 3.9 beschrieben.

Sicherstellung der Konsistenz Zur Wahrung der Konsistenz hinsichtlich der platzierten Annotationen werden diese als sogenannte indirekte Annotationen automatisch über Trace Links auf verbundene Elemente übertragen. Auf diese Weise ist bei Betrachtung eines skizzierten Elements, welches an mehreren Stellen im Projekt vorkommt, sofort ersichtlich, mit welchen Annotationen dieses insgesamt versehen wurde. Inkonsistenzen werden hierdurch verhindert, weil jedes Vorkommen des entsprechenden Elements überall mit den gleichen Annotationen versehen ist. Eine entsprechende Konsistenzprüfung wäre auch leicht für alle handgeschriebenen Texte möglich, findet jedoch im AugIR keine Anwendung, um den Gedanken- und Arbeitsfluss der Stakeholder während eines Workshops nicht zu stören. Dies wurde in Abschnitt 3.12.1 beschrieben.

Unterstützung für Identifikation von Widersprüchen Indirekte Annotationen unterstützen außerdem bei der Identifikation potentiell widersprüchlicher Interaction-Room-Annotationen. Diese sind in der Regel leicht zu erkennen, wenn sie auf demselben Element in derselben Skizze platziert wurden. Kommt das

gleiche Element jedoch in unterschiedlichen Skizzen vor und wird dort mit entsprechenden Annotationen versehen, so ist eine Identifikation der hierdurch entstehenden Widersprüche schwierig. Indirekte Annotationen helfen hierbei, indem sie alle für ein Element relevanten Annotationen gebündelt darstellen und darüber hinaus leicht automatisch ausgewertet werden können. Dies wurde ebenfalls in Abschnitt 3.12.1 beschrieben.

Unterstützung für Identifikation von Projektrisiken Darüber hinaus können sich aus den platzierten Annotationen mögliche Projektrisiken ableiten, die bei Verwendung analoger Zeichenwerkzeuge manuell identifiziert werden müssen. Der AugIR bietet die Möglichkeit, diese Annotationen transitiv über Trace Links auf ausgewählte Elemente zu aggregieren und dort zu sammeln. Dies erleichtert zum einen die manuelle Analyse der Elemente hinsichtlich potentieller Projektrisiken und erlaubt zum anderen eine automatische Auswertung. Dies wurde in Abschnitt 3.12.2 beschrieben.

Zusammenfassend lässt sich somit festhalten, dass alle in Abschnitt 1.3 aufgeführten Probleme prinzipiell durch den AugIR gelöst werden können. Dies bestätigt den ersten Teil der Forschungshypothese, die in Abschnitt 1.5 formuliert wurde. Dieser besagte, dass bei der gemeinschaftlichen Arbeit mit Freihandskizzen Einschränkungen und Nachteile aus der Verwendung analoger Zeichenwerkzeuge resultieren, die durch eine technische Augmentierung der eingesetzten Werkzeuge beseitigt werden können.

Darüber hinaus haben die durchgeführten Fallstudien und Experimente, die in Teil IV dieser Arbeit präsentiert wurden, gezeigt, dass der AugIR von fast allen Stakeholdern als nützliches Werkzeug wahrgenommen und als Ersatz für traditionelle analoge Whiteboards akzeptiert wird. Am Beispiel der Interaction-Room-Methode konnte gezeigt werden, dass die Verwendung digitaler Zeichenflächen zahlreiche Vorteile bietet, ohne dass sich hierbei negative Auswirkungen für die Stakeholder ergeben. Dies bestätigt den zweiten Teil der in Abschnitt 1.5 formulierten Forschungshypothese.

Insgesamt kann somit die Forschungshypothese, die dieser Arbeit zugrunde liegt, vollständig bestätigt werden.

10. Ausblick

Dieses abschließende Kapitel gibt einen Ausblick auf mögliche Forschungen, die sich an die Ergebnisse der vorliegenden Arbeit anschließen könnten.

10.1. Erweiterte Klassifizierung von Linienzügen

In Abschnitt 3.4 wurde ein Algorithmus entwickelt, der Linienzüge anhand ihrer charakteristischen Eigenschaften als Text- oder Form-Linienzüge klassifizieren kann. Obwohl bei der empirischen Bewertung der Features in Abschnitt 3.4.6 deutlich wurde, dass dieser für die vorliegende Datenbasis prinzipiell sehr gute Ergebnisse liefert und bereits 99,65 % aller Text-Linienzüge und 93,24 % aller Form-Linienzüge korrekt klassifizieren kann, wären weitere Optimierungen des Verfahrens denkbar.

Der vorgestellte Algorithmus funktioniert beispielsweise bisher nur auf Texten, die in Druckschrift verfasst wurden. Eine Klassifizierung von Schreibschrift ist nicht möglich, weil entsprechende Linienzüge gänzlich andere Charakteristiken aufweisen. Wenn ein Wort nicht mehr aus einzelnen Buchstaben zusammengesetzt ist sondern in einem durchgängigen Linienzug geschrieben wird, so ist dieser beispielsweise in der Regel deutlich länger und hat mehr Kontrollpunkte. Die Grenze zwischen Text und Form verschwimmt dadurch, weshalb viele der in Abschnitt 3.4 identifizierten Kriterien für eine Klassifizierung solcher Linienzüge nicht mehr geeignet sind. Es müssten deshalb andere Kriterien herangezogen werden, die beispielsweise eine komplexere und differenziertere Winkelbetrachtung vornehmen.

Allerdings ist fraglich, ob ein System realisierbar ist, welches Schreibschrift und Druckschrift gleichermaßen gut erkennen kann. Es ist anzunehmen, dass bestimmte

Kriterien einander widersprechen oder sich gegenseitig ausschließen und insbesondere unterschiedliche ideale Schwellwerte aufweisen. Dies würde zwangsläufig zu falsch erkannten Linienzügen führen, wenn innerhalb der gleichen Skizze Schreibschrift und Druckschrift vermischt werden.

Darüber hinaus haben Studien gezeigt, dass die Qualität der Klassifizierung von Linienzügen in Freihandskizzen mit anderen Verfahren, wie beispielsweise Support Vector Machines, gelegentlich besser ausfallen kann [54]. Obwohl bei der Evaluation des vorgestellten Ansatzes deutlich wurde, dass die logistische Regression für die vorliegende Datenbasis sehr gute Ergebnisse erzielt, wäre ein direkter Vergleich mit anderen Techniken spannend. Möglicherweise ließe sich hierdurch insbesondere die Erkennungsrate für Form-Linienzüge weiter verbessern, die im aktuellen Fall leicht unter der Erkennungsrate für Text-Linienzüge liegt.

Der vorgestellte Algorithmus könnte zudem um eine Mustererkennung erweitert werden, um die Ergebnisse der Klassifizierung noch weiter zu optimieren. *Tahuti* [97] verwendet beispielsweise eine komplexe Heuristik, die insbesondere Pfeilspitzen, welche aufgrund der Ähnlichkeit zu dem Buchstaben „v“ häufig als Text-Linienzüge erkannt werden, besser als Formen klassifizieren kann.

10.2. Prospektive Erfassung von Trace Links

Der in Abschnitt 3.8 vorgestellte Algorithmus zur Erfassung von Trace Links ist retrospektiv, weil er Verknüpfungen zwischen Texten nur aufgrund ihrer statischen Inhalte erfasst. Es wäre deshalb denkbar, ihn durch einen prospektiven Traceability-Algorithmus zu ergänzen, der darüber hinaus auch die Aktionen der Stakeholder überwacht und daraus geeignete Verknüpfungen ableitet.

Ein solcher Algorithmus könnte beispielsweise analysieren, welche Skizzen häufig in einem bestimmten Kontext betrachtet oder bearbeitet werden oder zwischen welchen Skizzen Stakeholder besonders oft navigieren. Hieraus könnten zusätzliche Kriterien für prospektive Trace Links abgeleitet werden. Darüber hinaus könnte eine solche

Analyse die Stärke bereits existierender Trace Links beeinflussen: Wenn Stakeholder beispielsweise überdurchschnittlich oft horizontal zwischen zwei verbundenen Elementen navigieren, so könnte dies die Stärke des Trace Links zwischen diesen erhöhen. Entsprechend könnte seine Stärke kontinuierlich abnehmen, wenn Stakeholder seit längerem nicht mehr zwischen den Elementen navigiert haben.

Im Hinblick auf die Verwendung prospektiver Traceability-Techniken wäre auch eine dynamische Anpassung des Schwellwerts ν für die Kosinus-Ähnlichkeit denkbar: Wenn Stakeholder über die Vorschauansicht häufig falsche Trace Links eliminieren, könnte dies ein Indiz für überdurchschnittlich viele False Positives sein. Eine dynamische Anhebung des Schwellwerts zur Laufzeit könnte in einem solchen Fall die Qualität der Trace-Link-Erfassung für das betrachtete Projekt verbessern, indem die Anzahl der falsch positiven Trace Links reduziert wird. Hierbei müsste jedoch darauf geachtet werden, den Schwellwert nicht zu weit anzuheben, weil ansonsten auch korrekte Trace Links verloren gehen.

Überdies könnte die Erfassung von inhaltlichen Zusammenhängen weiter verbessert werden, indem zusätzlich ein domänenspezifisches Wörterbuch und ein Thesaurus angebunden werden. Das Wörterbuch könnte bei der Korrektur von Handschrifterkennungsfehlern, die insbesondere bei Abkürzungen, Fachbegriffen und Eigennamen auftreten können, unterstützen. Es ließe sich sogar halbautomatisch aus bereits vorhandenen Dokumenten eines Projekts generieren. Ein Thesaurus würde darüber hinaus bei der Erfassung inhaltlicher Beziehungen zwischen semantisch ähnlichen Wörtern helfen. Beide könnten die Qualität der Trace-Link-Erfassung erhöhen und insbesondere den Recall verbessern.

10.3. Verteilter Interaction Room

Stakeholder eines Softwareprojekts arbeiten häufig räumlich getrennt voneinander an unterschiedlichen Standorten [2, 101]. Durch zunehmende Vereinfachung und Optimierung der zur Verfügung stehenden Kommunikationskanäle, die den Austausch von digitalen Artefakten ermöglichen, ist anzunehmen, dass die geografisch verteilte Entwicklung von Softwaresystemen auch in Zukunft weiter ansteigen wird [67].

Obwohl bei der Verwendung von Skizzen und Diagrammen in verteilten Entwicklungsteams bestimmte Barrieren existieren, spielen sie auch dort eine wichtige Rolle [259].

Nachdem das in dieser Arbeit vorgestellte Konzept eine Grundlage zur Erstellung digitaler Freihandskizzen auf elektronischen Whiteboards bietet, könnte im nächsten Schritt eine Variante des AugIRs konzipiert werden, die eine verteilte Durchführung von Interaction-Room-Workshops mit mehreren räumlich getrennten Gruppen ermöglicht.

Hierzu wäre insbesondere zu analysieren, welche Einschränkungen sich durch die räumliche Verteilung der teilnehmenden Stakeholder ergeben und wie diesen sowohl durch die AugIR-Software aber auch durch eine entsprechende Anpassung der Interaction-Room-Methode begegnet werden kann.

Es existieren bereits einige Ansätze für die gemeinschaftliche verteilte Erstellung digitaler Skizzen, die hierfür als Inspiration herangezogen werden könnten [92, 94, 129, 199].

10.4. Weitere Einsatzmöglichkeiten für mobile Endgeräte

Im AugIR kommen mobile Endgeräte zur Erstellung digitaler Karten für den Feature Canvas zum Einsatz. Die Diskussion verwandter Arbeiten in Kapitel 2 machte jedoch bereits deutlich, dass es darüber hinaus noch viele weitere sinnvolle Anwendungsmöglichkeiten für mobile Endgeräte geben kann.

Sie könnten beispielsweise als private Arbeitsflächen zur Erprobung experimenteller Ideen dienen. Obwohl Stakeholder häufig gemeinsam an einem bestimmten Problem arbeiten und dieses gemeinschaftlich am Whiteboard skizzieren und diskutieren [55], lösen sie sich auch gelegentlich von der Gruppe, um bestimmten Ideen alleine nachzugehen [92]. Der AugIR könnte diesen Stakeholdern deshalb die Möglichkeit geben, Inhalte vom Whiteboard auf ihr mobiles Endgerät zu transferieren, diese dort frei

zu bearbeiten oder weiterzuentwickeln und später die Ergebnisse zur Präsentation in der Gruppe auf das gemeinschaftlich genutzte Whiteboard zurück zu spielen.

Im Interaction-Room-Kontext erscheinen darüber hinaus insbesondere auch die beiden folgenden Optionen sinnvoll:

Stakeholder könnten die erstellten Skizzen direkt auf ihren mobilen Endgeräten mit Interaction-Room-Annotationen versehen. Dies hätte den Vorteil, dass beliebig viele Stakeholder gleichzeitig dieselbe Skizze annotieren könnten, ohne dass sie sich hierbei gegebenenfalls gegenseitig durch bereits platzierte Annotationen beeinflussen. Darüber hinaus könnte die Annotierung der Skizzen bequem aus der Entfernung erfolgen, ohne dass hierfür physisch mit den elektronischen Whiteboards interagiert werden müsste.

Weiterhin wäre es denkbar, Erläuterungen und Diskussionen zu Annotationen auf einem mobilen Endgerät zu erfassen, anstatt diese beispielsweise in einer separaten Tabelle zu protokollieren und manuell einen Verweis auf die entsprechende Nummer der zugehörigen Annotation hinzuzufügen. Die erfassten Texte könnten auf diese Weise direkt mit den Annotationen auf dem elektronischen Whiteboard verknüpft werden, was insbesondere bei der Workshop-Nachbereitung eine leichtere Zuordnung ermöglichen würde.

10.5. Zusätzliche Augmentierungen

Der Einsatz elektronischer Whiteboards und mobiler Endgeräte ist nur eine erste Möglichkeit, analoge Team-Räume sinnvoll durch Technik zu augmentieren. In naheliegender Weise könnten zusätzliche Geräte wie Kameras, Mikrofone oder Augmented-Reality-Brillen verwendet werden, um die Arbeit mit den digitalen Inhalten noch weiter zu vereinfachen und intuitiver zu gestalten.

Anstatt beispielsweise die Erläuterungen zu Annotationen schriftlich zu protokollieren, könnten diese von einem geeigneten Mikrofon aufgezeichnet und von einer Speech-To-Text-Engine automatisch in Text übersetzt werden. Die Aufzeichnung

ließe sich mit der zugehörigen Annotation verknüpfen und könnte beispielsweise beim Antippen derselben abgespielt oder als Text dargestellt werden.

Auch zusätzliche Sensoren könnten die Arbeit mit den elektronischen Whiteboards erleichtern und neue Interaktionsmöglichkeiten eröffnen. Ju et al. [116] präsentieren mit *Range* beispielsweise eine Anwendung für interaktive Whiteboards, die durch den Einsatz von Näherungssensoren auf das Verhalten der Stakeholder reagieren kann, ohne dass diese physisch mit dem Board interagieren müssen.

Überdies könnte durch eine zusätzliche Augmentierung des Raums auch die Erfassung inhaltlicher Beziehungen zwischen Skizzen weiter optimiert werden. Sharif und Kagdi [208] schlagen beispielsweise vor, durch den Einsatz von Okulographie (engl. Eye-Tracking) die Erfassung und Verwaltung von Trace Links zu verbessern.

Solche zusätzlichen technischen Erweiterungen könnten auch im AugIR Verwendung finden, um dessen Nützlichkeit für die Stakeholder noch weiter zu erhöhen.

A. Anhang

Der nachfolgende Anhang enthält ergänzende Detailinformationen, die jedoch zum Verständnis der Inhalte nicht zwingend erforderlich sind. Sie können bei Interesse nachgeschlagen werden und sind an entsprechender Stelle im Dokument referenziert.

A.1. Kosinusfunktion

An verschiedenen Stellen in dieser Arbeit wird die Kosinus-Funktion verwendet. Diese ist definiert als Abbildung $\cos : \mathbb{R} \to [-1, 1]$. Sie gehört zur Gruppe der trigonometrischen Funktionen und zeichnet sich durch einen periodischen Verlauf mit der Periode 2π aus.

Tabelle A.1 listet die wichtigsten Werte dieser Funktion als Übersicht auf.

Abbildung A.1 stellt ihren Verlauf im Intervall $[0, 2\pi]$ grafisch dar. Man sieht, dass die Kosinus-Funktion achsensymmetrisch zur y-Achse ist.

Tabelle A.1.: Winkeltabelle mit den wichtigsten Werten für die Kosinusfunktion.

Winkel	0°	45°	90°	135°	180°	225°	270°	315°	360°
Bogenmaß	0	$\frac{1}{4}\pi$	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$	π	$\frac{5}{4}\pi$	$\frac{3}{2}\pi$	$\frac{7}{4}\pi$	2π
$\cos x$	1	0,707...	0	-0,707...	-1	-0,707...	0	0,707...	1

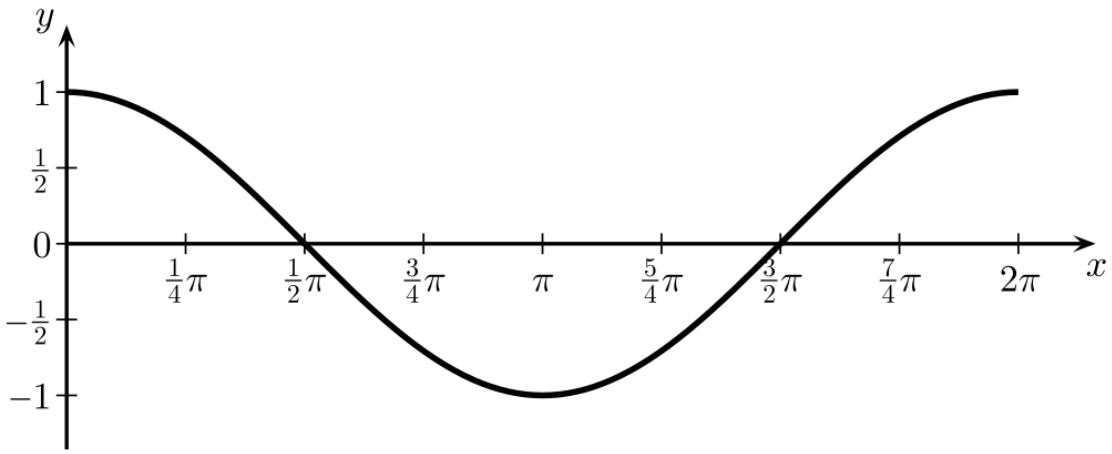

Abbildung A.1.: Grafischer Verlauf der Kosinusfunktion im Intervall $[0, 2\pi]$.

A.2. Detaillierte Messwerte zur Trace-Link-Erfassung

In Abschnitt 7.3.1 wurde bereits deutlich, dass sich für den Grenzwert $\nu = 0{,}55$ durchschnittlich die besten Ergebnisse für Precision und Recall erzielen lassen. Der Vollständigkeit halber erfolgt im Folgenden eine detaillierte individuelle Betrachtung der 16 Einzelprojekte:

Tabelle A.2 zeigt die Ergebnisse der Analyse bei Verwendung eines Schwellwerts von 0,9. Jede Zeile der Tabelle stellt die Resultate für jeweils ein Projekt dar. In der ersten Spalte sind die eindeutigen Kürzel der Projekte aufgeführt. Spalte 2 zeigt die Anzahl aller erfassten Trace Links für das entsprechende Projekt an. Dies umfasst sowohl korrekte Trace Links als auch False Positives. Die dritte Spalte stellt die Anzahl der korrekt erfassten Trace Links dar. Sie gibt also an, wie viele der erfassten Trace Links tatsächlich in den vorab erstellten Referenzlisten enthalten sind. In Spalte 4 ist die Anzahl der Verknüpfungen dargestellt, die vom Algorithmus erfasst wurden, obwohl sie nicht in den Referenzlisten aufgeführt sind. Spalte 5 zeigt an, wie viele Trace Links aus den Referenzlisten nicht vom Algorithmus gefunden wurden. Die Spalten 6 bis 8 stellen jeweils Precision, Recall und F für das entsprechende Projekt dar. Die letzten beiden Zeilen der Tabelle zeigen den akkumulierten Gesamtwert sowie den Durchschnittswert pro Projekt.

Tabelle A.2.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,9.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	39	39	0	3	1	0,93	0,96
P_2	24	24	0	14	1	0,63	0,77
P_3	1	1	0	9	1	0,1	0,18
P_4	20	18	2	0	0,9	1	0,95
P_5	41	37	4	13	0,9	0,74	0,81
P_6	16	14	2	5	0,88	0,74	0,8
P_7	39	38	1	32	0,97	0,54	0,7
P_8	44	44	0	1	1	0,98	0,99
P_9	14	14	0	3	1	0,82	0,9
P_{10}	31	29	2	13	0,94	0,69	0,79
P_{11}	2	2	0	8	1	0,2	0,33
P_{12}	26	24	2	8	0,92	0,75	0,83
P_{13}	10	10	0	21	1	0,32	0,49
P_{14}	61	61	0	28	1	0,69	0,81
P_{15}	6	5	1	12	0,83	0,29	0,43
P_{16}	106	100	6	93	0,94	0,52	0,67
Gesamt	480	460	20	263	0,96	0,64	0,76
Ø	30	28,75	1,25	16,44	0,96	0,62	0,71

Erwartungsgemäß ist die durchschnittliche Precision mit 0,96 sehr hoch, weil es aufgrund des hohen Schwellwerts für die Kosinus-Ähnlichkeit nur wenige falsch positive Trace Links gibt. Bei der Hälfte aller Projekte liegt die Precision sogar bei 1, d. h. dort sind alle erfassten Trace Links korrekt. Außer bei den beiden Projekten P_6 und P_{15} ist die Precision durchgängig größer oder gleich 90 %.

Demgegenüber ist der durchschnittliche Recall mit 0,64 wie erwartet sehr niedrig. Für einen Kosinus-Ähnlichkeitsschwellwert von 0,9 werden folglich viele der geschätzten Trace Links übersehen. Lediglich bei Projekt P_4 werden bereits 100 % aller Trace Links aus den Referenzlisten erfasst. Projekt P_8 erreicht mit 0,98 ebenfalls einen sehr guten Recall. Für alle anderen Projekte ist dieser aber deutlich zu niedrig.

Der Wert des F-Maßes liegt wie eingangs geschätzt dazwischen und hat einen durchschnittlichen Wert von 0,71.

Um den Recall zu erhöhen, muss der Schwellwert ν für die Kosinus-Ähnlichkeit weiter abgesenkt werden. Jedoch darf er dabei nicht zu niedrig gewählt werden, um ein zu starkes Abfallen der Precision zu vermeiden.

Da die Resultate im Bereich 0,9 bis 0,75 wie in Abbildung 7.1 dargestellt relativ konstant bleiben, wird als nächster interessanter Schwellwert die Grenze 0,7 betrachtet. Tabelle A.3 illustriert die entsprechenden Werte für alle Projekte für diesen Kosinus-Ähnlichkeitsschwellwert.

Die durchschnittliche Precision fällt um 0,03 von 0,96 auf 0,93. Während der Wert für die meisten Projekte unverändert hoch bleibt, kommen bei einigen wenigen Projekten durch die Absenkung des Schwellwerts neue falsch positive Trace Links hinzu, was sich negativ auf die Precision auswirkt. Am deutlichsten wird dies bei den Projekten P_{13} und P_{15}. Bei P_{13} kommen fünf neue False Positives hinzu, was die Precision um 0,21 von 1 auf 0,79 absinken lässt. Bei P_{15} steigt die Zahl der falsch positiven Trace Links auf sieben an, was für einen Abfall der Precision um 0,2 von 0,83 auf 0,63 sorgt. Insgesamt kommen für alle Projekte zusammen 18 neue False Positives hinzu, wodurch sich die Gesamtzahl der falsch positiven Trace Links auf 38 erhöht.

Tabelle A.3.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,7.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	39	39	0	3	1	0,93	0,96
P_2	24	24	0	14	1	0,63	0,77
P_3	7	7	0	3	1	0,7	0,82
P_4	20	18	2	0	0,9	1	0,95
P_5	43	39	4	11	0,91	0,78	0,84
P_6	16	14	2	5	0,88	0,74	0,8
P_7	61	60	1	10	0,98	0,86	0,92
P_8	44	44	0	1	1	0,98	0,99
P_9	14	14	0	3	1	0,82	0,9
P_{10}	33	31	2	11	0,93	0,74	0,83
P_{11}	2	2	0	8	1	0,2	0,33
P_{12}	26	24	2	8	0,92	0,75	0,83
P_{13}	24	19	5	12	0,79	0,61	0,69
P_{14}	83	80	3	9	0,96	0,9	0,93
P_{15}	19	12	7	5	0,63	0,71	0,67
P_{16}	165	155	10	38	0,93	0,8	0,87
Gesamt	620	582	38	141	0,94	0,8	0,87
Ø	38,75	36,38	2,38	8,81	0,93	0,76	0,82

Demgegenüber steigt jedoch der durchschnittliche Recall deutlich um 0,14 von 0,62 auf 0,76, weil durch die Absenkung von ν auf 0,7 für einige Projekte viele der zuvor übersehenen Trace Links nun korrekt erkannt werden. Beispielsweise erhöht sich der Recall für Projekt P_7 um 0,32 von 0,54 auf 0,86. Für P_{15} steigt er sogar um 0,42 von 0,29 auf 0,71. Insgesamt sinkt die Gesamtzahl der übersehenen Trace Links dadurch von 263 auf 141.

Entsprechend erhöht sich auch der Wert des F-Maßes, welcher für $\nu = 0{,}7$ nun durchschnittlich 0,82 beträgt und damit um 0,11 gegenüber $\nu = 0{,}9$ ansteigt. Insgesamt ist somit die Qualität der erfassten Trace Links für den Schwellwert 0,7 bereits deutlich besser als für 0,9.

Aus Abbildung 7.1 ist ersichtlich, dass ein weiteres Absenken des Kosinus-Ähnlichkeitsschwellwerts eine zusätzliche Verbesserung der Resultate verspricht. Tabelle A.4 zeigt zunächst die Ergebnisse für einen Schwellwert von 0,6.

Wie man sieht, ändern sich die durchschnittlichen Werte für Precision, Recall und F bei der Absenkung des Schwellwerts von 0,7 auf 0,6 nur geringfügig: Während die durchschnittliche Precision unverändert bei 0,93 liegt, steigt der durchschnittliche Recall leicht um 0,02 von 0,76 auf 0,78. Der Wert des F-Maßes erhöht sich ebenfalls geringfügig von 0,82 auf 0,83. Für viele Projekte bleiben die entsprechenden Werte somit unverändert. Die größte Veränderung erfährt Projekt P_{15}: Hier wächst der Recall um 0,11 von 0,71 auf 0,82, während gleichzeitig auch die Precision um 0,04 von 0,63 auf 0,67 steigt.

Auf den ersten Blick mag es verwunderlich erscheinen, dass die Precision ansteigt obwohl der Schwellwert für die Kosinus-Ähnlichkeit gesenkt wird. Dieses Phänomen lässt sich jedoch wie folgt erklären: Wenn durch das Absenken des Schwellwerts zusätzliche Trace Links erfasst werden und es sich bei diesen neu hinzukommenden Verknüpfungen um korrekte Trace Links statt False Positives handelt, so können diese die Precision positiv beeinflussen. Dies wird deutlich, wenn man Projekt P_{15} näher betrachtet: Durch eine Absenkung des Schwellwerts von 0,7 auf 0,6 hat sich die Anzahl der erfassten Trace Links für dieses Projekt um 2 von 19 auf 21 erhöht. Weil beide neu erfassten Trace Links korrekt sind, steigt folglich ebenfalls die Anzahl

Tabelle A.4.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,6.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	39	39	0	3	1	0,93	0,96
P_2	24	24	0	14	1	0,63	0,77
P_3	7	7	0	3	1	0,7	0,82
P_4	20	18	2	0	0,9	1	0,95
P_5	46	42	4	8	0,91	0,84	0,88
P_6	16	14	2	5	0,88	0,74	0,8
P_7	61	60	1	10	0,98	0,86	0,92
P_8	44	44	0	1	1	0,98	0,99
P_9	14	14	0	3	1	0,82	0,9
P_{10}	35	33	2	9	0,94	0,79	0,86
P_{11}	2	2	0	8	1	0,2	0,33
P_{12}	28	26	2	6	0,93	0,81	0,87
P_{13}	24	19	5	12	0,79	0,61	0,69
P_{14}	85	82	3	7	0,96	0,92	0,94
P_{15}	21	14	7	3	0,67	0,82	0,74
P_{16}	171	160	11	33	0,94	0,83	0,88
Gesamt	637	598	39	125	0,94	0,83	0,88
Ø	39,81	37,38	2,44	7,81	0,93	0,78	0,83

Tabelle A.5.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,55.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	44	40	4	2	0,91	0,95	0,93
P_2	24	24	0	14	1	0,63	0,77
P_3	8	8	0	2	1	0,8	0,89
P_4	20	18	2	0	0,9	1	0,95
P_5	49	45	4	5	0,92	0,9	0,91
P_6	16	14	2	5	0,88	0,74	0,8
P_7	67	66	1	4	0,99	0,94	0,96
P_8	44	44	0	1	1	0,98	0,99
P_9	16	16	0	1	1	0,94	0,97
P_{10}	41	39	2	3	0,95	0,93	0,94
P_{11}	3	3	0	7	1	0,3	0,46
P_{12}	33	31	2	1	0,94	0,97	0,95
P_{13}	32	27	5	4	0,84	0,87	0,86
P_{14}	88	85	3	4	0,97	0,96	0,96
P_{15}	22	15	7	2	0,68	0,88	0,77
P_{16}	195	176	19	17	0,9	0,91	0,91
Gesamt	702	651	51	72	0,93	0,9	0,91
Ø	43,88	40,69	3,19	4,5	0,93	0,86	0,88

der korrekt erfassten Trace Links um 2 von 12 auf 14 an. Man rechnet nun leicht mit Hilfe der Formeln aus Abschnitt 7.2 nach, dass $\frac{12}{19} \approx 0{,}6316 < 0{,}6667 \approx \frac{14}{21}$ gilt.

Tabelle A.5 zeigt die Resultate, die sich bei einem erneuten Absenken des Kosinus-Ähnlichkeitsschwellwerts von 0,6 auf 0,55 ergeben. Während die durchschnittliche Precision konstant bei 0,93 verharrt, kann der durchschnittliche Recall noch einmal deutlich um 0,08 von 0,78 auf 0,86 gesteigert werden. Dementsprechend erhöht sich auch der durchschnittliche Wert des F-Maßes von 0,83 auf 0,88.

Abbildung 7.1 hat bereits angedeutet, dass eine weitere Verbesserung der Werte durch ein erneutes Absenken des Schwellwerts nicht möglich ist. Diese Vermutung wird durch die Zahlen in Tabelle A.6 bestätigt, welche die entsprechenden Werte für

Tabelle A.6.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,525.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	44	40	4	2	0,91	0,95	0,93
P_2	24	24	0	14	1	0,63	0,77
P_3	8	8	0	2	1	0,8	0,89
P_4	20	18	2	0	0,9	1	0,95
P_5	49	45	4	5	0,92	0,9	0,91
P_6	16	14	2	5	0,88	0,74	0,8
P_7	68	66	2	4	0,99	0,94	0,96
P_8	44	44	0	1	1	0,98	0,99
P_9	16	16	0	1	1	0,94	0,97
P_{10}	41	39	2	3	0,95	0,93	0,94
P_{11}	3	3	0	7	1	0,3	0,46
P_{12}	33	31	2	1	0,94	0,97	0,95
P_{13}	32	27	5	4	0,84	0,87	0,86
P_{14}	88	85	3	4	0,97	0,96	0,96
P_{15}	24	15	9	2	0,68	0,88	0,77
P_{16}	195	176	19	17	0,9	0,91	0,91
Gesamt	705	651	54	72	0,92	0,9	0,91
Ø	44,06	40,69	3,38	4,5	0,93	0,86	0,88

einen Kosinus-Ähnlichkeitsschwellwert von 0,525 angibt.

Man sieht, dass alle drei Werte für Precision, Recall und F stagnieren. Die Precision beträgt weiterhin 0,93 während der Recall bei 0,86 liegt und der Wert des F-Maßes 0,88 beträgt. Aus Spalte 2 ist jedoch ersichtlich, dass für $\nu = 0{,}525$ insgesamt drei Trace Links mehr erfasst werden als für $\nu = 0{,}55$. Spalte 4 zeigt, dass alle diese Trace Links False Positives sind, weshalb die durchschnittliche Precision leicht um 0,4 % abfällt. Dies ist jedoch aus der Tabelle aufgrund der Rundung auf zwei Nachkommastellen nicht unmittelbar ersichtlich. Die Resultate für einen Kosinus-Ähnlichkeitsschwellwert von 0,525 sind somit marginal schlechter als für 0,55.

Aus Abbildung 7.1 war bereits ersichtlich, dass die Precision beim Übergang auf

Tabelle A.7.: Analyse von Precision, Recall und F-Maß für einen Kosinus-Ähnlichkeitsschwellwert von 0,5.

Projekt	Erfasste TLs	Korrekte TLs	False Positives	Übersehene TLs	Precision	Recall	F
P_1	82	42	40	0	0,51	1	0,68
P_2	30	30	0	8	1	0,79	0,88
P_3	18	10	8	0	0,56	1	0,7
P_4	20	18	2	0	0,9	1	0,95
P_5	114	47	67	3	0,41	0,94	0,57
P_6	18	14	4	5	0,78	0,74	0,76
P_7	72	67	5	3	0,93	0,96	0,94
P_8	95	44	51	1	0,46	0,98	0,63
P_9	29	17	12	0	0,59	1	0,74
P_{10}	45	41	4	1	0,91	0,98	0,94
P_{11}	4	3	1	7	0,75	0,3	0,43
P_{12}	45	31	14	1	0,69	0,97	0,81
P_{13}	63	30	33	1	0,48	0,97	0,64
P_{14}	99	86	13	3	0,87	0,97	0,91
P_{15}	42	17	25	0	0,4	1	0,58
P_{16}	274	180	94	13	0,66	0,93	0,77
Gesamt	1 050	677	373	46	0,64	0,94	0,76
Ø	65,63	42,31	23,31	2,88	0,68	0,91	0,75

einen Schwellwert von 0,5 schlagartig signifikant abfällt. Dies wird durch Tabelle A.7 verdeutlicht: Obwohl der durchschnittliche Recall noch einmal um 0,05 von 0,86 auf 0,91 gesteigert werden kann, fällt die Precision drastisch um 0,25 von 0,93 auf 0,68 ab. Die Genauigkeit sinkt somit um 25 Prozentpunkte. Dies spiegelt sich auch in F wieder, welches ebenfalls deutlich um 0,13 von 0,88 auf 0,75 abfällt.

Obwohl 0,91 ein sehr guter Recall für den vorgestellten Traceability-Algorithmus wäre, ist eine Precision von 0,68 unverhältnismäßig niedrig. Beim Übergang von 0,525 auf 0,5 hat sich die Anzahl der falsch positiven Trace Links von 54 auf 373 erhöht und somit fast versiebenfacht. Die Wahl von 0,5 als Schwellwert lässt sich somit keinesfalls rechtfertigen, da die Anzahl der False Positives unakzeptabel hoch ist.

Wie in Abschnitt 7.3.1 diskutiert wird somit $nu = 0{,}55$ als endgültiger Schwellwert für die Trace-Link-Erfassung gewählt.

A.3. Statistische Analyse der Bearbeitungszeit

In diesem Abschnitt werden die Zeiten analysiert, die die Teilnehmer der in Kapitel 8 beschriebenen Studie zur Lösung der Aufgaben benötigten. Hierzu wird eine Varianzanalyse mit Messwiederholung durchgeführt [30]. Die Faktoren für diese Analyse sind die jeweilige Gruppe des betrachteten Teilnehmers, die bearbeitete Aufgabe, sowie das Projekt, anhand dessen die Aufgabe bearbeitet wurde. Für den Faktor *Gruppe* werden die beiden Ausprägungen „Gruppe 1“ und „Gruppe 2“ betrachtet. Der Faktor *Projekt* liegt in den Ausprägungen „Projekt 1“ und „Projekt 2“ vor. Für den Faktor *Aufgabe* existieren entsprechend die Ausprägungen „Aufgabe 1“ und „Aufgabe 2“. Wie im vorangegangenen Abschnitt beschrieben, erfolgte in Gruppe 1 die Bearbeitung der Aufgaben für Projekt 1 im Interaction Room und für Projekt 2 im AugIR. Umgekehrt erfolgte in Gruppe 2 die Bearbeitung der Aufgaben für Projekt 1 im AugIR und für Projekt 2 im Interaction Room.

Im Zuge der folgenden statistischen Auswertung werden zunächst die Signifikanzen und Effektstärken der Faktoren bzw. Faktorkombinationen sowie eventuelle Zwischen- und Innersubjekteffekte untersucht. Daraufhin werden anhand der zugehörigen Interaktionsdiagramme mögliche Interaktionseffekte betrachtet und diskutiert. Abschließend werden statistische Tests zur detaillierten Analyse der Messreihen durchgeführt.

A.3.1. Analyse der Signifikanzen und Effektstärken

Zunächst wurde eine Varianzanalyse mit Messwiederholung durchgeführt. Tabelle A.8 zeigt die Ergebnisse der Analyse der Faktoren hinsichtlich Signifikanz und Effektstärke.

Tabelle A.8.: Ergebnis der Varianzanalyse mit Messwiederholung der Faktoren hinsichtlich Signifikanz und Effektstärke.

Faktor(en)	p	η_p^2
Gruppe	0,615	0,013
Projekt	0,004	0,344
Aufgabe	<0,001	0,843
Gruppe $*$ Projekt	<0,001	0,967
Gruppe $*$ Aufgabe	0,495	0,024
Projekt $*$ Aufgabe	0,006	0,321
Gruppe $*$ Projekt $*$ Aufgabe	0,256	0,064

Die erste Spalte führt den bzw. die jeweilige(n) Faktor(en) auf. Zunächst werden alle Faktoren einzeln betrachtet, bevor deren Kombinationen untersucht werden. Eine Kombination mehrerer Faktoren wird als $Faktor_1 * Faktor_2$ notiert.

Die zweite Spalte zeigt die p-Werte [246] der zugehörigen Faktoren. Der p-Wert gibt eine Wahrscheinlichkeit an und liegt deshalb zwischen 0 und 1. Er gibt Auskunft darüber, ob ein Faktor bzw. eine Faktorkombination statistisch signifikant ist. Dies kann dann angenommen werden, wenn p kleiner als 0,05 ist [254].

Spalte 3 listet die entsprechenden Werte für das partielle Eta-Quadrat [194] auf. Dieses wird im folgenden als η_p^2 notiert und zeigt die Stärke eines statistischen Effekts als Wert zwischen 0 und 1 an. Je größer das partielle Eta-Quadrat ist, desto stärker ist der Einfluss des entsprechenden Faktors bzw. der Faktorkombination.

Zunächst wird untersucht, ob ein Zwischensubjekteffekt vorliegt, also ob der Zwischensubjektfaktor *Gruppe* ein signifikanter Faktor ist. Wie die erste Zeile in Tabelle A.8 zeigt, ist dies wegen $p = 0{,}615$ nicht der Fall. Selbst wenn *Gruppe* ein signifikanter Faktor wäre, so wäre seine Effektstärke sehr gering, weil das zugehörige partielle Eta-Quadrat einen Wert von $\eta_p^2 = 0{,}013$ aufweist.

Als nächstes werden mögliche Innersubjekteffekte untersucht. *Projekt* ist mit $p = 0{,}004$ ein signifikanter Faktor. Zudem ist sein Einfluss nicht unerheblich, weil das zugehörige partielle Eta-Quadrat $\eta_p^2 = 0{,}344$ beträgt. Ebenso ist *Aufgabe* mit $p <$

0,001 ein signifikanter Faktor. Dieser hat zudem aufgrund des hohen partiellen Eta-Quadrats von $\eta_p^2 = 0{,}843$ einen sehr großen Einfluss.

Im nächsten Schritt werden mögliche Interaktionseffekte betrachtet. Tabelle A.8 zeigt, dass zwischen *Gruppe* und *Projekt* wegen $p < 0{,}001$ ein starker Interaktionseffekt vorliegt. Zudem ist die Effektstärke aufgrund des entsprechenden partiellen Eta-Quadrats von $\eta_p^2 = 0{,}967$ sehr groß.

Demgegenüber existiert kein Interaktionseffekt zwischen *Gruppe* und *Aufgabe* ($p = 0{,}495$ und $\eta_p^2 = 0{,}024$).

Jedoch gibt es wegen $p = 0{,}006$ einen Interaktionseffekt zwischen *Projekt* und *Aufgabe*. Die entsprechende Effektstärke ist mit einem partiellen Eta-Quadrat von $\eta_p^2 = 0{,}321$ nicht unerheblich.

Es kann kein Interaktionseffekt zwischen allen drei Faktoren *Gruppe*, *Projekt* und *Aufgabe* festgestellt werden ($p = 0{,}256$ und $\eta_p^2 = 0{,}064$).

Zusammenfassend lässt sich somit festhalten, dass zwischen den Faktoren *Gruppe* und *Projekt* ein sehr starker Interaktionseffekt vorliegt. Des Weiteren existiert ein nicht unerheblicher Interaktionseffekt zwischen den Faktoren *Projekt* und *Aufgabe*. Um die Art der Interaktionseffekte genauer zu untersuchen, werden im Folgenden die entsprechenden Interaktionsdiagramme betrachtet.

A.3.2. Analyse der Interaktionen

Abbildung A.2 zeigt die beiden Interaktionsdiagramme für die Faktoren *Gruppe* und *Projekt*. Es wird deutlich, dass zwischen diesen eine disordinale Interaktion vorliegt, weil sich die entsprechenden Linien in beiden Interaktionsdiagrammen schneiden [30]. Dies bedeutet, dass durch die Wahl eines Parameters nur in Kombination mit dem jeweils anderen die abhängige Variable – in diesem Fall die zur Lösung der entsprechenden Aufgabe benötigte Zeit – beeinflusst wird. Die benötigte Zeit hängt somit von der Kombination der Faktoren *Gruppe* und *Projekt* ab.

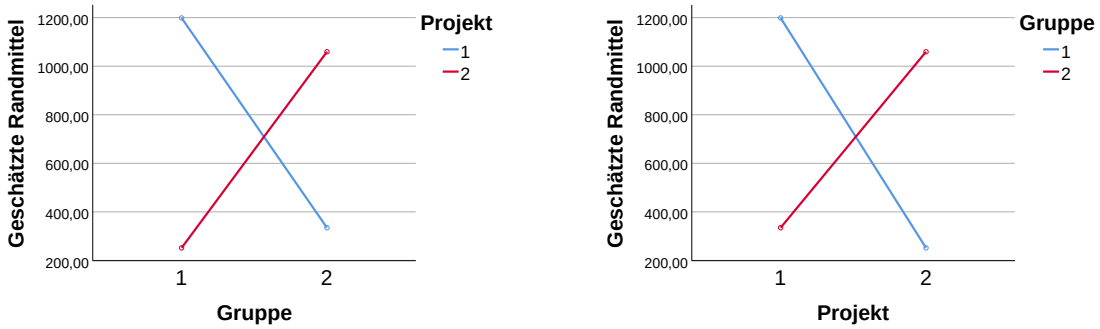

Abbildung A.2.: Die Interaktionsdiagramme für die Faktoren *Gruppe* und *Projekt* zeigen eine disordinale Interaktion.

Darüber hinaus wird aus den Interaktionsdiagrammen deutlich, dass alle Teilnehmer in beiden Gruppen sämtliche Aufgaben im AugIR durchgängig schneller lösen konnten als im Interaction Room.

Abbildung A.3 zeigt die beiden Interaktionsdiagramme für die Faktoren *Projekt* und *Aufgabe*. Anders als im vorherigen Fall liegt hier eine ordinale Interaktion vor, weil sich die Linien in den entsprechenden Interaktionsdiagrammen nicht schneiden [30].

Aus den Interaktionsdiagrammen lassen sich mehrere Erkenntnisse ablesen: In beiden Projekten benötigten die Probanden zur Lösung von Aufgabe 1 jeweils mehr Zeit als zur Lösung von Aufgabe 2. Weiterhin variiert der zeitliche Unterschied zwischen Aufgabe 1 und Aufgabe 2 in Abhängigkeit davon, ob diese in Projekt 1 oder Projekt 2 bearbeitet wurden. Die Bearbeitung der Aufgaben in Projekt 1 dauerte jeweils durchschnittlich länger als die Bearbeitung derselben Aufgaben in Projekt 2. Dies gilt insbesondere für die erste Aufgabe. Während diese in Projekt 1 durchschnittlich mehr Zeit in Anspruch nahm als in Projekt 2, konnte Aufgabe 2 in beiden Projekten im Durchschnitt annähernd gleich schnell bearbeitet werden.

Insgesamt lassen sich die bisherigen Erkenntnisse wie folgt zusammenfassen und interpretieren:

Die Wahl der Gruppe ist für das Resultat des Experiments nicht entscheidend. Die Gruppenzugehörigkeit eines Probanden hat also grundsätzlich keinen Einfluss auf

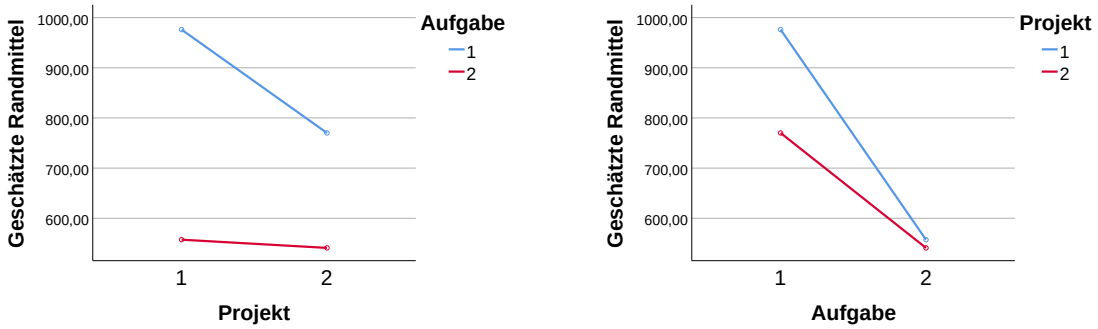

Abbildung A.3.: Die Interaktionsdiagramme für die Faktoren *Projekt* und *Aufgabe* zeigen eine ordinale Interaktion.

das Gesamtergebnis.

Die zur Lösung einer Aufgabe benötigte Zeit wird durch die Gruppe des Teilnehmers in Kombination mit dem bearbeiteten Projekt bestimmt. Dies ist aus der disordinalen Interaktion der entsprechenden Faktoren ersichtlich. Das ist naheliegend, wenn man bedenkt, dass beide Gruppen zwar die gleichen Projekte bearbeitet haben, die Gruppenzugehörigkeit jedoch darüber entschied, welches Projekt im AugIR und welches im Interaction Room bearbeitet wurde.

Die Aufgaben sind unterschiedlich, was sich in der benötigten Zeit widerspiegelt. Die Bearbeitung von Aufgabe 1 nahm durchschnittlich mehr Zeit in Anspruch als die Bearbeitung der zweiten Aufgabe. Es ist somit sinnvoll, im Folgenden beide Aufgaben getrennt voneinander zu betrachten.

Ebenso sind die Projekte unterschiedlich. Die Lösung der Aufgaben nahm für Projekt 1 durchschnittlich mehr Zeit in Anspruch als für Projekt 2. Aus diesem Grund werden im Folgenden auch die Projekte getrennt voneinander betrachtet.

Darüber hinaus wurde aus den Interaktionsdiagrammen deutlich, dass die Faktoren *Aufgabe* und *Projekt* interagieren. Deshalb werden diese Faktoren im Folgenden ebenfalls getrennt voneinander analysiert.

Tabelle A.9.: Ergebnisse des Kolmogorov-Smirnov-Tests auf Normalverteilung.

Gruppe / Aufgabe / Projekt	Signifikanz mit Kolmogorov-Smirnov-Test
Gruppe 1 / Aufgabe 1 / Projekt 1	0,2
Gruppe 1 / Aufgabe 1 / Projekt 2	0,084
Gruppe 1 / Aufgabe 2 / Projekt 1	0,085
Gruppe 1 / Aufgabe 2 / Projekt 2	0,2
Gruppe 2 / Aufgabe 1 / Projekt 1	0,2
Gruppe 2 / Aufgabe 1 / Projekt 2	0,2
Gruppe 2 / Aufgabe 2 / Projekt 1	0,2
Gruppe 2 / Aufgabe 2 / Projekt 2	0,2

A.3.3. Analyse getrennt nach Treatments

Um eine getrennte Betrachtung der Ausprägungen vorzunehmen, können vier ungepaarte T-Tests durchgeführt werden [225]. Die Tests sind ungepaart, da für jeden Probanden pro Aufgabe und Projekt nur jeweils ein Messwert vorliegt. Eine Voraussetzung für die Durchführung dieser Tests ist jedoch eine Normalverteilung aller Messwerte des gleichen Zwischensubjektfaktors, d. h. alle Messwerte von Gruppe 1 und alle Messwerte von Gruppe 2 müssen jeweils normalverteilt sein.

Um dies zu untersuchen, wurde für alle Kombinationen der Kolmogorov-Smirnov-Test (kurz KS-Test) [51] durchgeführt. Die Ergebnisse sind in Tabelle A.9 dargestellt. Die erste Spalte zeigt jeweils die Kombination aus *Gruppe*, *Aufgabe* und *Projekt*. Die zweite Spalte enthält die zugehörige Signifikanz des KS-Tests. Ein Wert von 0,2 zeigt an, dass die entsprechende Messreihe normalverteilt ist. Bei $p < 0{,}2$ liegt keine Normalverteilung vor.

Wie man sieht, sind alle Messreihen von Gruppe 2 normalverteilt. Für Gruppe 1 gilt dies jedoch nicht: Die Werte für Aufgabe 1 und Projekt 2 sowie für Aufgabe 2 und Projekt 1 sind wegen $p = 0{,}084$ bzw. $p = 0{,}085$ nicht normalverteilt. Hieraus resultiert, dass nur für Aufgabe 1 / Projekt 1 sowie für Aufgabe 2 / Projekt 2 ein T-Test durchgeführt werden darf. Für die anderen beiden Messreihen Aufgabe 2 / Projekt 1 sowie Aufgabe 1 / Projekt 2 muss ein nicht-parametrischer Test durchge-

Tabelle A.10.: Resultate des T-Tests für alle Messreihen für den Faktor *Gruppe*.

Aufgabe / Projekt	Sign.	Mittl. Differenz	95 % Konfidenzintervall	
Aufgabe 1 / Projekt 1	< 0,001	921,82	757,81	1 085,82
Aufgabe 1 / Projekt 2	< 0,001	822,09	629,16	1 015,02
Aufgabe 2 / Projekt 1	< 0,001	806,18	676,22	936,15
Aufgabe 2 / Projekt 2	< 0,001	793,45	603,18	983,73

führt werden, der nicht auf einer Normalverteilungsannahme basiert. Hierfür wird der populäre Mann-Whitney-U-Test gewählt [158].

Tabelle A.10 zeigt zunächst die Ergebnisse des T-Tests. Obwohl es wie zuvor beschrieben formal eigentlich nicht korrekt ist, diesen für nicht normalverteilte Messreihen durchzuführen, enthält Tabelle A.10 dennoch die Resultate des Tests für alle Messreihen, damit auf diese später nach Durchführung des Mann-Whitney-U-Tests Bezug genommen werden kann.

Spalte 1 der Tabelle zeigt die in der jeweiligen Zeile betrachtete Kombination aus *Aufgabe* und *Projekt*. Die zweite Spalte gibt die Signifikanz des T-Tests an. In der dritten Spalte ist die mittlere Differenz aufgeführt. Diese gibt die Differenz der Mittelwerte zwischen beiden Gruppen an. Spalten 4 und 5 zeigen die untere bzw. obere Grenze des 95 % Konfidenzintervalls. Dieses beschreibt, welche Mittelwertdifferenzen zwischen den betrachteten Gruppen erwartet würden, wenn die vorliegende Studie unendlich oft wiederholt werden würde. Je kleiner die Abstände zwischen oberer und unterer Grenze sind, desto genauer kann der Unterschied zwischen den Mittelwerten der beiden Gruppen auf Basis der bereits gesammelten Daten vorhergesagt werden.

Der T-Test für Aufgabe 1 und Projekt 1 ist signifikant mit $p < 0{,}001$. Die mittlere Differenz liegt bei ca. 921 Sekunden. Die Probanden, die Aufgabe 1 für das erste Projekt im AugIR bearbeiteten, waren somit durchschnittlich ca. 921 Sekunden bzw. 15 Minuten und 21 Sekunden schneller, als die Probanden, die die gleiche Aufgabe im Interaction Room bearbeiteten. Das 95 % Konfidenzintervall liegt zwischen 757 und 1085 Sekunden.

Tabelle A.11.: Ergebnisse des Mann-Whitney-U-Tests.

Gruppe / Aufgabe / Projekt	Signifikanz	Mittlerer Rang
Gruppe 1 / Aufgabe 1 / Projekt 2	$< 0{,}001$	6
Gruppe 2 / Aufgabe 1 / Projekt 2	$< 0{,}001$	17
Gruppe 1 / Aufgabe 2 / Projekt 1	$< 0{,}001$	17
Gruppe 2 / Aufgabe 2 / Projekt 1	$< 0{,}001$	6

Der T-Test für Aufgabe 2 und Projekt 2 ist ebenfalls signifikant mit $p < 0{,}001$. Hier ergibt sich eine mittlere Differenz von ca. 793 Sekunden. Die Teilnehmer benötigten somit zur Bearbeitung der zweiten Aufgabe in Projekt 2 im AugIR durchschnittlich ca. 793 Sekunden bzw. 13 Minuten und 13 Sekunden weniger als im Interaction Room. Das 95 % Konfidenzintervall liegt zwischen 603 und 983 Sekunden.

Weil für die Kombinationen Aufgabe 1 und Projekt 2 sowie für Aufgabe 2 und Projekt 1 wie zuvor beschrieben streng genommen kein T-Test durchgeführt werden darf, müssen die entsprechenden Messwerte mit einem anderen Verfahren ausgewertet werden. Hierzu wird der Mann-Whitney-U-Test verwendet. Die entsprechenden Resultate sind in Tabelle A.11 aufgeführt. Spalte 1 zeigt die jeweilige Kombination der Faktoren *Gruppe*, *Aufgabe* und *Projekt* an. Spalte 2 zeigt die entsprechende Signifikanz und in Spalte 3 ist der zugehörige mittlere Rang aufgeführt. Dieser gibt die Position des zugehörigen Messwerts innerhalb der Liste aller nach ihrer Größe sortierten und durchnummerierten Messwerte an.

Die erste Zeile zeigt, dass der mittlere Rang für die Kombination Gruppe 1 / Aufgabe 1 / Projekt 2 den Wert 6 besitzt. Aus Zeile 2 geht hingegen hervor, dass für die Kombination Gruppe 2 / Aufgabe 1 / Projekt 2 der mittlere Rang 17 beträgt. Dies bedeutet, dass Teilnehmer der ersten Gruppe zur Bearbeitung der ersten Aufgabe in Projekt 2 weniger Zeit benötigten als Teilnehmer der zweiten Gruppe. Weil die Probanden aus Gruppe 1 die Aufgabe im AugIR und die Probanden aus Gruppe 2 die Aufgabe im Interaction Room bearbeitet haben, bedeutet dies folglich, dass die Bearbeitung der Aufgabe im AugIR in kürzerer Zeit erfolgte als im Interaction Room.

Entsprechend sieht man in Zeile 3, dass der mittlere Rang für die Kombination

Gruppe 1 / Aufgabe 2 / Projekt 1 den Wert 17 besitzt. Zeile 4 zeigt hingegen einen mittleren Rang von 6 für die Kombination Gruppe 2 / Aufgabe 2 / Projekt 1. Dies bedeutet, dass die Teilnehmer der zweiten Gruppe zur Bearbeitung der zweiten Aufgabe in Projekt 1 weniger Zeit benötigten als Teilnehmer der ersten Gruppe. Weil die Probanden aus Gruppe 2 die Aufgabe im AugIR und die Teilnehmer der ersten Gruppe die Aufgabe im Interaction Room bearbeitet haben, bedeutet dies, dass die Bearbeitung im AugIR erneut schneller war als im Interaction Room.

Auch wenn die Werte für die vorgenannten Messreihen nicht normalverteilt sind und deshalb eigentlich kein T-Test durchgeführt werden darf, so stützen die Resultate des Mann-Whitney-U-Tests dennoch die Ergebnisse aus Tabelle A.10. Dort ist zu sehen, dass die mittlere Differenz für Aufgabe 1 und Projekt 2 bei ca. 822 Sekunden liegt. Die Bearbeitung der ersten Aufgabe in Projekt 2 erfolgt im AugIR somit durchschnittlich ca. 822 Sekunden bzw. 13 Minuten und 42 Sekunden schneller als im Interaction Room. Das 95 % Konfidenzintervall liegt zwischen 629 und 1015 Sekunden. Die mittlere Differenz für Aufgabe 2 und Projekt 1 liegt bei ca. 806 Sekunden. Die zweite Aufgabe in Projekt 1 konnte somit im AugIR durchschnittlich ca. 13 Minuten und 26 Sekunden schneller bearbeitet werden als im Interaction Room. Das 95 % Konfidenzintervall liegt zwischen 676 und 936 Sekunden.

A.3.4. Interpretation der Ergebnisse

Insgesamt lassen sich die zuvor präsentierten Ergebnisse wie folgt interpretieren und zusammenfassen:

Zunächst kann festgehalten werden, dass die Gruppenzugehörigkeit eines Probanden keinen Einfluss auf das Gesamtergebnis des Experiments hatte, weil der Faktor *Gruppe* nicht signifikant ist.

Weiterhin ist aus der Analyse der Interaktionseffekte ersichtlich, dass die Faktoren *Gruppe* und *Aufgabe* frei miteinander kombinierbar sind, ohne dass dies einen Einfluss auf das Gesamtergebnis des Experiments hat.

Es wurde deutlich, dass die Bearbeitung von Aufgabe 1 durchschnittlich mehr Zeit in Anspruch nahm als die Bearbeitung von Aufgabe 2. Dies könnte ein Indiz dafür sein, dass die Identifikation von textuellen Zusammenhängen schwieriger oder aufwändiger ist als das Auffinden von Annotationsnummern.

Überdies dauerte die Bearbeitung der Aufgaben in Projekt 1 durchschnittlich etwas länger als die Bearbeitung der Aufgaben in Projekt 2. Dies könnte aus der Komplexität der Projekte resultieren, da Projekt 1 umfangreicher ist als Projekt 2.

Die durchgeführten T-Tests haben gezeigt, dass die Bearbeitung von Aufgabe 1 in Projekt 1 und von Aufgabe 2 in Projekt 2 im AugIR jeweils wesentlich schneller möglich war als im Interaction Room. Ein ähnliches Ergebnis folgte aus den Mann-Whitney-U-Tests für die Bearbeitungszeiten von Aufgabe 1 in Projekt 2 und Aufgabe 2 in Projekt 1.

Durch das Experiment konnte somit gezeigt werden, dass Stakeholder inhaltliche Beziehungen zwischen Skizzen im AugIR schneller erkennen können als auf traditionellen analogen Whiteboards. Die ermittelten Ergebnisse waren dabei statistisch signifikant. Insgesamt bestätigt dies die eingangs aufgestellte Hypothese. Es lässt sich folglich festhalten, dass der AugIR die Stakeholder bei der schnellen Identifikation von inhaltlichen Zusammenhängen unterstützt.

A.4. Statistische Analyse der Fehler

Bereits in Abschnitt 8.5 wurde deutlich, dass die Teilnehmer des in Kapitel 8 präsentierten Experiments bei der Bearbeitung der Aufgaben im Interaction Room wesentlich mehr Fehler machten als im AugIR. Dennoch wird der Form und Vollständigkeit halber im Folgenden ergänzend eine formale Analyse der Ergebnisse mit Mitteln der deskriptiven Statistik vorgenommen, indem eine Varianzanalyse mit Messwiederholung durchgeführt wird [30]. Das Vorgehen ist dabei vollständig analog zur Analyse der benötigten Bearbeitungszeit im vorangegangenen Abschnitt: Zunächst werden die Signifikanzen der Faktoren bzw. Faktorkombiniationen sowie

Tabelle A.12.: Ergebnis der Varianzanalyse mit Messwiederholung der Faktoren hinsichtlich Signifikanz und Effektstärke.

Faktor(en)	p	η_p^2
Gruppe	0,221	0,074
Projekt	0,221	0,074
Aufgabe	0,022	0,237
Gruppe $*$ Projekt	<0,001	0,825
Gruppe $*$ Aufgabe	<0,001	0,513
Projekt $*$ Aufgabe	<0,001	0,513
Gruppe $*$ Projekt $*$ Aufgabe	0,022	0,237

eventuelle Zwischen- und Innersubjekteffekte untersucht. Daraufhin werden mögliche Interaktionseffekte betrachtet und diskutiert. Abschließend wird ein T-Test zur Analyse der Messreihen durchgeführt.

A.4.1. Analyse der Signifikanzen und Effektstärken

Tabelle A.12 zeigt die Analyse der Faktoren und Faktorkombinationen hinsichtlich ihrer Signifikanz und Effektstärke. Die erste Spalte listet die verschiedenen Faktoren bzw. Faktorkombinationen auf. Spalte 2 stellt die zugehörigen p-Werte dar und aus Spalte 3 ist das entsprechende partielle Eta-Quadrat ersichtlich.

Der Faktor *Gruppe* ist nicht signifikant ($p = 0{,}221$ und $\eta_p^2 = 0{,}074$). Demnach liegt kein Zwischensubjekteffekt vor.

Der Faktor *Projekt* ist ebenfalls nicht signifikant ($p = 0{,}221$ und $\eta_p^2 = 0{,}074$).

Der Faktor *Aufgabe* ist hingegen wegen $p = 0{,}022$ signifikant. Das zugehörige partielle Eta-Quadrat hat einen Wert von $\eta_p^2 = 0{,}237$, weshalb die Effektstärke dieses Faktors zudem nicht unerheblich ist.

Zur Identifikation möglicher Interaktionseffekte müssen darüber hinaus die Faktorkombinationen betrachtet werden. Es existiert ein sehr starker Interaktionseffekt zwischen *Gruppe* und *Projekt*, weil die Signifikanz dieser Kombination einen Wert

von $p < 0{,}001$ aufweist. Die Effektstärke ist aufgrund des hohen partiellen Eta-Quadrats von $\eta_p^2 = 0{,}825$ sehr groß. Darüber hinaus existiert ebenfalls jeweils ein mittelgroßer Interaktionseffekt zwischen *Gruppe* und *Aufgabe* sowie zwischen *Projekt* und *Aufgabe*. In beiden Fällen liegt jeweils ein p-Wert von $p < 0{,}001$ und ein partielles Eta-Quadrat von $\eta_p^2 = 0{,}513$ vor. Anders als bei der Betrachtung der benötigten Bearbeitungszeit kann in diesem Fall wegen $p = 0{,}022$ darüber hinaus ein Interaktionseffekt zwischen allen drei Faktoren *Gruppe*, *Projekt* und *Aufgabe* beobachtet werden. Die Effektstärke ist wegen $\eta_p^2 = 0{,}237$ nicht unerheblich.

Zusammenfassend lässt sich somit festhalten, dass für die Faktorkombinationen *Gruppe* $*$ *Projekt*, *Gruppe* $*$ *Aufgabe*, *Projekt* $*$ *Aufgabe* und *Gruppe* $*$ *Projekt* $*$ *Aufgabe* jeweils teils starke Interaktionseffekte vorliegen. Um die Art der Interaktionseffekte genauer zu untersuchen, werden im Folgenden die zugehörigen Interaktionsdiagramme betrachtet.

A.4.2. Analyse der Interaktionen

Abbildung A.4 zeigt die beiden Interaktionsdiagramme für die Faktoren *Gruppe* und *Projekt*. Es liegt eine disordinale Interaktion vor. Die Fehleranzahl hängt somit von der Kombination der Faktoren *Gruppe* und *Projekt* ab. Aus den Diagrammen ist überdies ersichtlich, dass beide Gruppen bei der Bearbeitung der Aufgaben im Interaction Room jeweils mehr Fehler machten als im AugIR. Insbesondere wird deutlich, dass alle Teilnehmer die Aufgaben im AugIR vollständig fehlerfrei lösen konnten, während dies im Interaction Room nicht möglich war.

Abbildung A.5 zeigt die Interaktionsdiagramme für die Faktoren *Gruppe* und *Aufgabe*. Auch in diesem Fall liegt eine disordinale Interaktion vor. Die Fehleranzahl hängt somit ebenfalls von der Kombination der Faktoren *Gruppe* und *Aufgabe* ab. Darüber hinaus ist aus den Diagrammen ersichtlich, dass bei Aufgabe 1 von den Teilnehmern beider Gruppen in Summe durchschnittlich mehr Fehler gemacht wurden als bei Aufgabe 2.

Abbildung A.6 zeigt die Interaktionsdiagramme für die Faktoren *Projekt* und *Aufgabe*. Erneut liegt eine disordinale Interaktion vor. Folglich hängt die Fehlerzahl somit

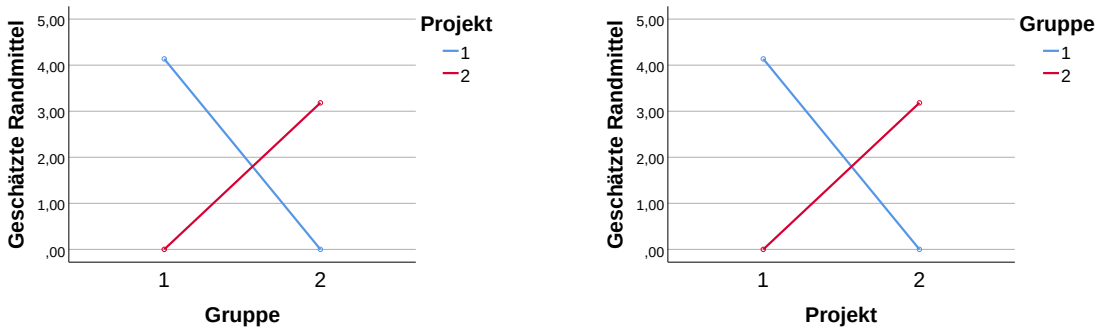

Abbildung A.4.: Die Interaktionsdiagramme für die Faktoren *Gruppe* und *Projekt* zeigen eine disordinale Interaktion.

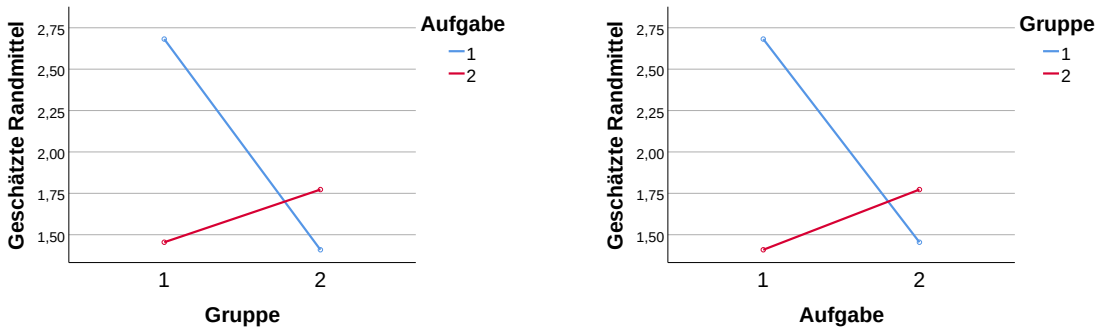

Abbildung A.5.: Die Interaktionsdiagramme für die Faktoren *Gruppe* und *Aufgabe* zeigen eine disordinale Interaktion.

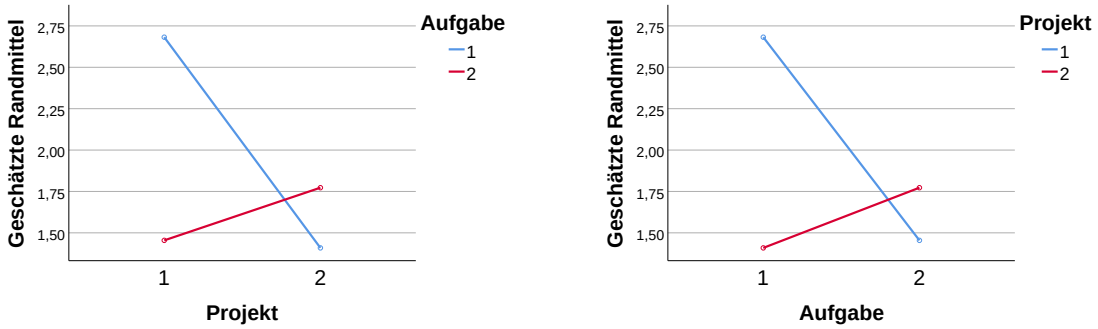

Abbildung A.6.: Die Interaktionsdiagramme für die Faktoren *Projekt* und *Aufgabe* zeigen eine disordinale Interaktion.

ebenfalls von der Kombination der Faktoren *Projekt* und *Aufgabe* ab. Man sieht, dass in Projekt 1 bei der Bearbeitung von Aufgabe 1 mehr Fehler gemacht wurden als bei der Bearbeitung von Aufgabe 2, während dies in Projekt 2 genau umgekehrt war. Insgesamt wurden bei Aufgabe 1 mehr Fehler gemacht als bei Aufgabe 2.

Der Vollständigkeit halber müssen auch die Interaktionsdiagramme der Faktorkombination *Gruppe* $*$ *Projekt* $*$ *Aufgabe* betrachtet werden, da zwischen allen drei Faktoren ebenfalls ein starker Interaktionseffekt existiert. Hierzu müssen die Interaktionsdiagramme für je zwei Faktoren separat analysiert werden, wobei jeweils der dritte Faktor als Konstante behandelt wird.

Die entsprechenden Interaktionsdiagramme sind in den Abbildungen A.7 bis A.12 dargestellt. Sie liefern jedoch keine neuen Erkenntnisse sondern bestätigen lediglich die bisherigen Beobachtungen. Um den Rahmen dieses Abschnitts nicht zu sprengen wird deshalb auf eine ausführliche Beschreibung verzichtet.

Im nachfolgenden Abschnitt wird abschließend erneut eine detaillierte statistische Analyse der Messwerte vorgenommen.

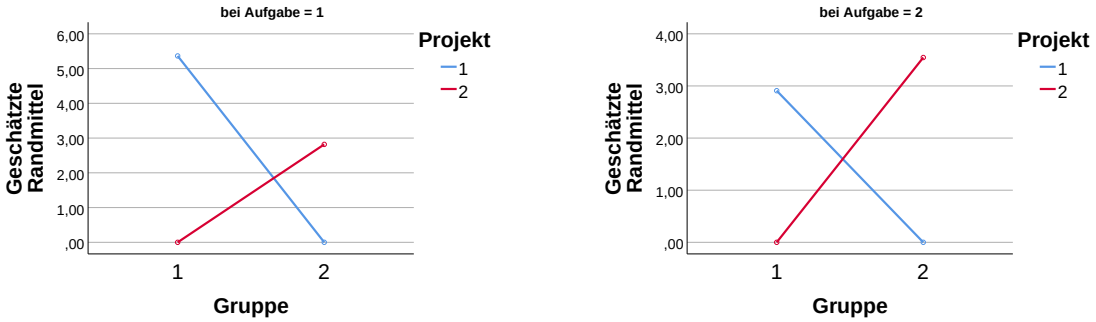

Abbildung A.7.: Die Interaktionsdiagramme für die Faktorkombination *Gruppe* ∗ *Projekt* zeigen eine disordinale Interaktion. Der dritte Faktor *Aufgabe* wird jeweils als Konstante behandelt.

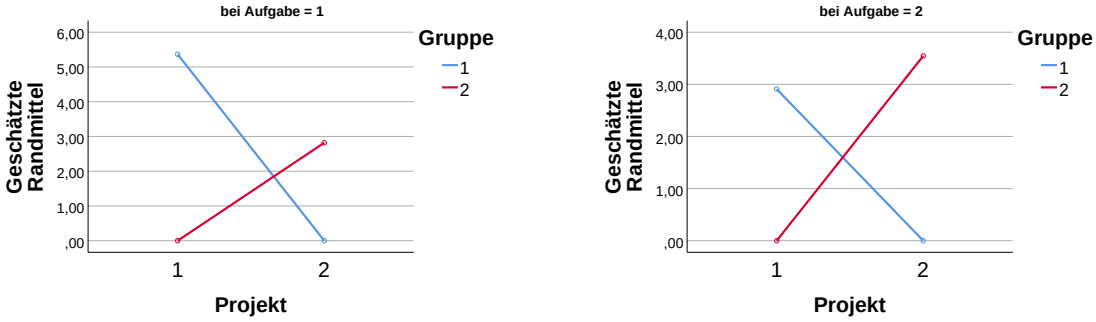

Abbildung A.8.: Die Interaktionsdiagramme für die Faktorkombination *Projekt* ∗ *Gruppe* zeigen eine disordinale Interaktion. Der dritte Faktor *Aufgabe* wird jeweils als Konstante behandelt.

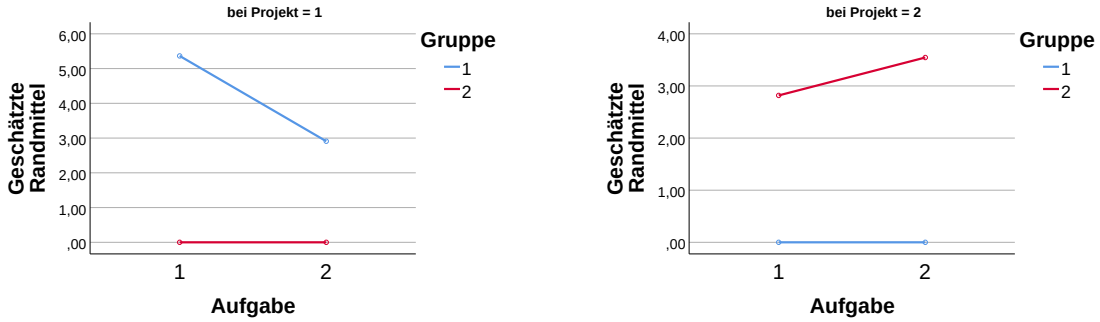

Abbildung A.9.: Die Interaktionsdiagramme für die Faktorkombination *Aufgabe* ∗ *Gruppe* zeigen eine ordinale Interaktion. Der dritte Faktor *Projekt* wird jeweils als Konstante behandelt.

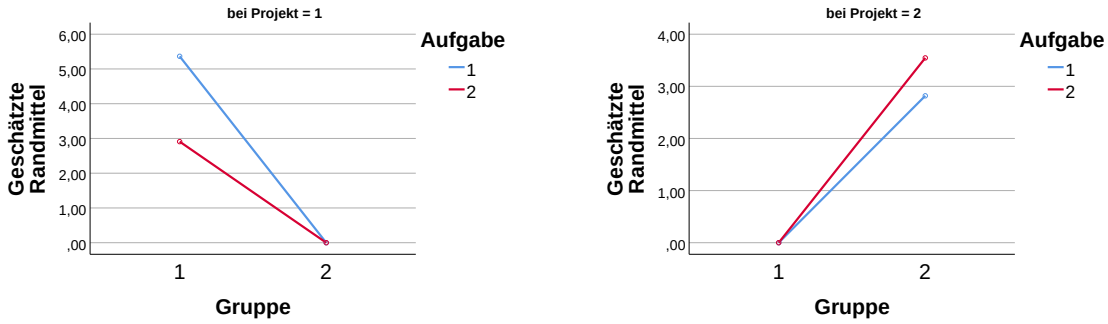

Abbildung A.10.: Die Interaktionsdiagramme für die Faktorkombination *Gruppe* ∗ *Aufgabe* zeigen eine ordinale Interaktion. Der dritte Faktor *Projekt* wird jeweils als Konstante behandelt.

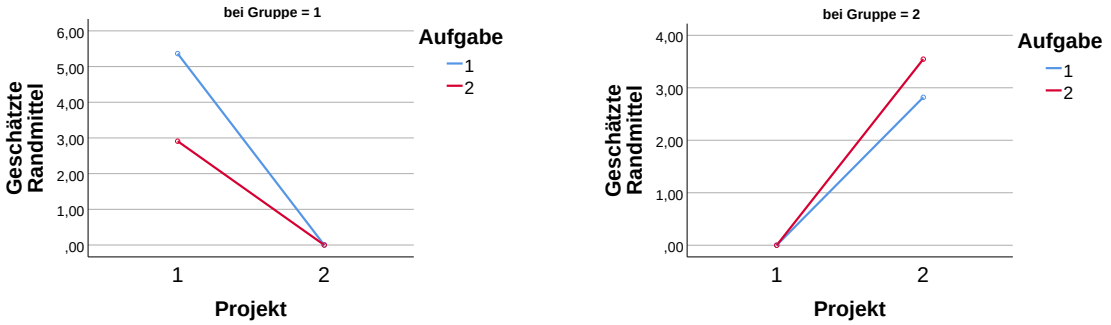

Abbildung A.11.: Die Interaktionsdiagramme für die Faktorkombination *Projekt* ∗ *Aufgabe* zeigen eine ordinale Interaktion. Der dritte Faktor *Gruppe* wird jeweils als Konstante behandelt.

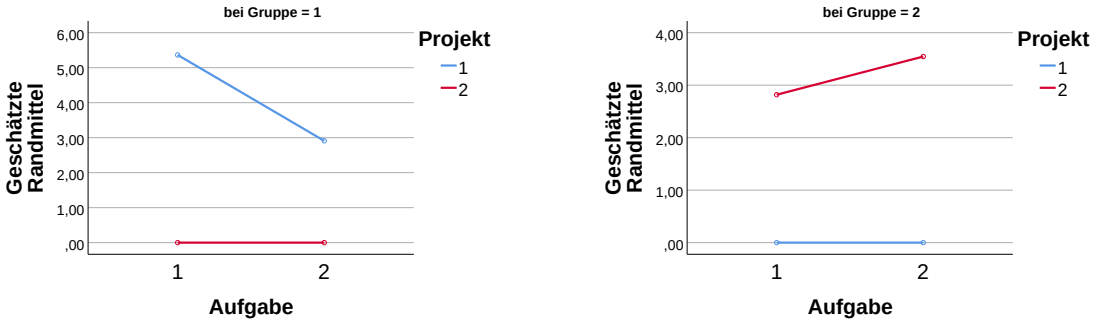

Abbildung A.12.: Die Interaktionsdiagramme für die Faktorkombination *Aufgabe* ∗ *Projekt* zeigen eine ordinale Interaktion. Der dritte Faktor *Gruppe* wird jeweils als Konstante behandelt.

Tabelle A.13.: Resultate des T-Tests gegen 0 für alle Messreihen für den Faktor *Gruppe*.

Aufgabe / Projekt	Sign.	Mittl. Differenz	95 % Konfidenzintervall	
Aufgabe 1 / Projekt 1	$< 0{,}001$	5,36	3,91	6,81
Aufgabe 1 / Projekt 2	$< 0{,}001$	2,82	1,86	4,34
Aufgabe 2 / Projekt 1	$< 0{,}001$	2,91	1,99	3,83
Aufgabe 2 / Projekt 2	$< 0{,}001$	3,55	2,03	5,06

A.4.3. Analyse getrennt nach Treatments

Weil kein einziger Teilnehmer bei der Bearbeitung der Aufgaben im AugIR einen Fehler machte, ist die Fehlerzahl für alle betreffenden Probanden konstant gleich 0. Aus diesem Grund kann im Folgenden ein T-Test gegen 0 durchgeführt werden. Das Ergebnis ist aus Tabelle A.13 ersichtlich. Die erste Spalte zeigt die jeweils betrachtete Kombination aus *Aufgabe* und *Projekt*. Spalte 2 listet die jeweilige Signifikanz auf. In Spalte 3 ist die mittlere Differenz angegeben, während die vierte und fünfte Spalte den unteren und oberen Wert des 95 % Konfidenzintervalls auflisten.

Zunächst ist festzuhalten, dass alle Tests signifikant sind mit $p < 0{,}001$.

Erwartungsgemäß entspricht die mittlere Differenz aller Tests den bereits in den Tabellen 8.2 und 8.3 angegebenen Mittelwerten, weil die Fehlerzahl für die Bearbeitung der Aufgaben im AugIR konstant 0 entspricht.

Bei der Bearbeitung von Aufgabe 1 in Projekt 1 machte jeder Teilnehmer somit im Interaction Room durchschnittlich 5,36 Fehler mehr als im AugIR. Das 95 % Konfidenzintervall liegt zwischen 3,91 und 6,81.

Bei der Bearbeitung von Aufgabe 1 in Projekt 2 machte jeder Teilnehmer im Interaction Room durchschnittlich 2,82 Fehler mehr als im AugIR. Das 95 % Konfidenzintervall liegt zwischen 1,86 und 4,34.

Bei der Bearbeitung von Aufgabe 2 in Projekt 1 machte jeder Teilnehmer im Interaction Room durchschnittlich 2,91 Fehler mehr als im AugIR. Das 95 % Konfidenzintervall liegt zwischen 1,99 und 3,83.

Bei der Bearbeitung von Aufgabe 2 in Projekt 2 machte jeder Teilnehmer im Interaction Room durchschnittlich 3,55 Fehler mehr als im AugIR. Das 95 % Konfidenzintervall liegt zwischen 2,03 und 5,06.

A.4.4. Interpretation der Ergebnisse

Auch bei der Betrachtung der gemachten Fehler hat die Gruppenzugehörigkeit eines Probanden keinen Einfluss auf das Gesamtergebnis des Experiments. Dies folgt erneut daraus, dass *Gruppe* kein signifikanter Faktor ist. Die Wahl der Gruppe ist somit für das Resultat des Experiments nicht entscheidend.

Die Anzahl der Fehler, die von den Probanden bei der Lösung der Aufgaben gemacht wurden, bestimmt sich durch eine Kombination aller betrachteten Faktoren. Dies ist jeweils aus der disordinalen Interaktion der entsprechenden Faktoren ersichtlich.

Während beide Aufgaben in beiden Projekten im AugIR vollständig fehlerfrei gelöst werden konnten, war dies im Interaction Room nicht möglich. Bei der Bearbeitung der ersten Aufgabe traten hierbei in Summe durchschnittlich mehr Fehler auf als bei der Bearbeitung der zweiten Aufgabe. Dies stützt die bereits in Abschnitt A.3.4 geäußerte Vermutung, dass die fehlerfreie Identifikation von textuellen Zusammenhängen schwieriger ist als das Auffinden von Annotationsnummern.

Hinsichtlich der Projekte konnte beobachtet werden, dass die Teilnehmer bei der Bearbeitung von Aufgabe 1 in Projekt 1 im Interaction Room durchschnittlich fast doppelt so viele Fehler machten wie in Projekt 2. Demgegenüber ist jedoch die Anzahl der gemachten Fehler bei der Bearbeitung von Aufgabe 2 in Projekt 1 im Interaction Room etwas niedriger als in Projekt 2. Im AugIR wurden in beiden Projekten keinerlei Fehler gemacht.

Insgesamt konnte somit durch das Experiment gezeigt werden, dass Stakeholder im AugIR seltener relevante inhaltliche Zusammenhänge übersehen als auf traditionellen analogen Whiteboards. Die ermittelten Ergebnisse waren dabei statistisch signifikant. Dies bestätigt die eingangs formuliert Hypothese. Es lässt sich folglich

festhalten, dass der AugIR die Stakeholder bei der fehlerfreien Aufdeckung inhaltlicher Beziehungen unterstützt.

Literaturverzeichnis

[1] K. Alisch, E. Winter, and U. Arentzen, *Gabler Wirtschafts Lexikon*. Springer-Verlag, 2013.

[2] J. Altmann and G. Pomberger, *Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge*. Heidelberg: Physica-Verlag HD, 1999, pp. 643–664. [Online]. Available: https://doi.org/10.1007/978-3-642-58663-7_34

[3] S. Ambler, *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. Wiley, 2002.

[4] C. Andrews, A. Endert, and C. North, "Space to think: large high-resolution displays for sensemaking," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 55–64.

[5] M. Andric, W. Hall, and L. Carr, "Assisting artifact retrieval in software engineering projects," in *Proceedings of the 2004 ACM Symposium on Document Engineering*, ser. DocEng '04. New York, NY, USA: ACM, 2004, pp. 48–50. [Online]. Available: http://doi.acm.org/10.1145/1030397.1030407

[6] C. Anslow, S. Marshall, J. Noble, and R. Biddle, "Co-located collaborative software visualization," in *Human Aspects of Software Engineering*, ser. HAoSE '10. New York, NY, USA: ACM, 2010, pp. 4:1–4:2. [Online]. Available: http://doi.acm.org/10.1145/1938595.1938603

[7] ——, "Interactive multi-touch surfaces for software visualization," in *Proceedings of the Workshop on Data Exploration for Interactive Surfaces DEXIS 2011*, 2011, pp. 20–23.

[8] C. Anslow, S. Marshall, J. Noble, E. Tempero, and R. Biddle, "User evaluation of polymetric views using a large visualization wall," in *Proceedings of the 5th international symposium on Software visualization*, ser. SOFTVIS '10. New York, NY, USA: ACM, 2010, pp. 25–34.

[9] L. Anthony and J. O. Wobbrock, "A lightweight multistroke recognizer for user interface prototypes," in *Proceedings of Graphics Interface 2010*, ser. GI '10. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2010, pp. 245–252.

[10] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Trans. Softw. Eng.*, vol. 28, no. 10, pp. 970–983, Oct. 2002.

[11] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 95–104.

[12] H. U. Asuncion and R. N. Taylor, "Capturing custom link semantics among heterogeneous artifacts and tools," in *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, ser. TEFSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–5.

[13] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM press New York, 1999, vol. 463.

[14] B. P. Bailey and J. A. Konstan, "Are informal tools better? comparing demais, pencil and paper, and authorware for early multimedia design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '03. New York, NY, USA: ACM, 2003, pp. 313–320. [Online]. Available: http://doi.acm.org/10.1145/642611.642666

[15] A. Baker, A. van der Hoek, H. Ossher, and M. Petre, "Guest editors' introduction: Studying professional software design," *IEEE Software*, vol. 29, no. 1, pp. 28–33, Jan 2012.

[16] R. Ball and C. North, "The effects of peripheral vision and physical navigation on large scale visualization," in *Proceedings of graphics interface 2008*, ser. GI '08. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2008, pp. 9–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1375714.1375717

[17] R. Ball, C. North, and D. A. Bowman, "Move to improve: promoting physical navigation to increase user performance with large displays," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 191–200.

[18] S. Baltes and S. Diehl, "Sketches and diagrams in practice," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 530–541. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635891

[19] A. Begel, N. Nagappan, C. Poile, and L. Layman, "Coordination in large-scale software teams," in *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, ser. CHASE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/CHASE.2009.5071401

[20] J. Behnke, *Logistische Regressionsanalyse: Eine Einführung.* Springer-Verlag, 2014.

[21] E. Behrends, *Elementare Stochastik.* Vieweg+Teubner Verlag, 2012.

[22] R. Bellamy, M. Desmond, J. Martino, P. Matchen, H. Ossher, J. Richards, and C. Swart, "Sketching tools for ideation (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 808–811. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985909

[23] X. Bi and R. Balakrishnan, "Comparing usage of a large high-resolution display to single or dual desktop displays for daily work," in *Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1005–1014.

[24] C. M. Bishop, M. Svensen, and G. E. Hinton, "Distinguishing text from graphics in on-line handwritten ink," in *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, ser. IWFHR '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 142–147. [Online]. Available: https://doi.org/10.1109/IWFHR.2004.34

[25] R. Blagojevic, S. H.-H. Chang, and B. Plimmer, "The power of automatic feature selection: Rubine on steroids," in *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, ser. SBIM '10. Eurographics Association, 2010, pp. 79–86.

[26] M. Book, S. Grapenthin, and V. Gruhn, "Risk-aware migration of legacy data structures," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, 2013, pp. 53–56.

[27] ——, "Seeing the forest and the trees: focusing team interaction on value and effort drivers," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE '12. New York, NY, USA: ACM, 2012, pp. 30:1–30:4.

[28] ——, "Value-based migration of legacy data structures," in *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering*, ser. Lecture Notes in Business Information Processing, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Springer International Publishing, 2014, vol. 166, pp. 115–134.

[29] M. Book, V. Gruhn, and R. Striemer, *Tamed Agility - Pragmatic Contracting and Collaboration in Agile Software Projects*. Springer International Publishing, 2016.

[30] J. Bortz, *Statistik für Sozialwissenschaftler*. Springer, 1999.

[31] A. Bozzon, M. Brambilla, and P. Fraternali, "Searching repositories of web application models," in *Proceedings of the 10th International Conference on Web Engineering*, ser. ICWE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=1884110.1884112

[32] P. Brandl, M. Haller, J. Oberngruber, and C. Schafleitner, "Bridging the gap between real printouts and digital whiteboard," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '08. New York, NY, USA: ACM, 2008, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/1385569.1385577

[33] Buckley, Chris and Salton, Gerard, "Stop Word List 2," 2018, [accessed 2018-02-09]. [Online]. Available: http://www.lextek.com/manuals/onix/stopwords2.html

[34] D. J. Burr, "Designing a handwriting reader," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 5, pp. 554–559, May 1983. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1983.4767435

[35] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "A system of patterns: Pattern-oriented software architecture," 1996.

[36] B. Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[37] A. Caetano, N. Goulart, M. Fonseca, and J. Jorge, "Javasketchit: Issues in sketching the look of user interfaces," in *AAAI Spring Symposium on Sketch Understanding*. AAAI Press, Menlo Park, 2002, pp. 9–14.

[38] C. Calhoun, T. F. Stahovich, T. Kurtoglu, and L. B. Kara, "Recognizing multi-stroke symbols," in *AAAI Spring Symposium on Sketch Understanding*, 2002, pp. 15–23.

[39] Q. Chen, J. Grundy, and J. Hosking, "An e-whiteboard application to support early design-stage sketching of uml diagrams," in *Human Centric Computing*

Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on, 2003, pp. 219–226.

[40] ——, "Sumlow: early design-stage sketching of uml diagrams on an e-whiteboard," *Software - Practice & Experience*, vol. 38, no. 9, pp. 961–994, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1002/spe.v38:9

[41] X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 223–232.

[42] X. Chen, J. Hosking, and J. Grundy, "A combination approach for enhancing automated traceability (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 912–915.

[43] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: how and why software developers use drawings," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 557–566. [Online]. Available: http://doi.acm.org/10.1145/1240624.1240714

[44] N. A. Chinchor and B. Sundheim, "Message understanding conference (muc) tests of discourse processing," in *Proc. AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, 1995, pp. 21–26.

[45] R. Chung, P. Mirica, and B. Plimmer, "Inkkit: A generic design tool for the tablet pc," in *Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-human Interaction: Making CHI Natural*, ser. CHINZ '05. New York, NY, USA: ACM, 2005, pp. 29–30. [Online]. Available: http://doi.acm.org/10.1145/1073943.1073950

[46] J. Cleland-Huang, O. Gotel, and A. Zisman, *Software and Systems Traceability*. Springer, 2012.

[47] L. M. Covi, J. S. Olson, E. Rocco, W. J. Miller, and P. Allie, "A room of your own: What do we learn about support of teamwork from assessing teams in dedicated project rooms," in *Proceedings of Cooperative Buildings: Integrating Information, Organization and Architecture: First International Workshop, Co'Build '98*. ACM Press, 1998.

[48] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[49] A. Coyette, J. Vanderdonckt, and Q. Limbourg, "Sketchixml: A design tool for informal user interface rapid prototyping," in *Rapid Integration of Software Engineering Techniques*, ser. Lecture Notes in Computer Science, N. Guelfi and D. Buchs, Eds. Springer Berlin Heidelberg, 2007, vol. 4401, pp. 160–176. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71876-5_11

[50] C. H. Damm, K. M. Hansen, and M. Thomsen, "Tool support for cooperative object-oriented design: Gesture based modelling on an electronic whiteboard," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ser. CHI '00. New York, NY, USA: ACM, 2000, pp. 518–525. [Online]. Available: http://doi.acm.org/10.1145/332040.332488

[51] W. W. Daniel, *Applied Nonparametric Statistics*. Wadsworth Pub Co, 1989.

[52] T. de Souza Alcantara, J. Ferreira, and F. Maurer, "Interactive prototyping of tabletop and surface applications," in *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '13. New York, NY, USA: ACM, 2013, pp. 229–238. [Online]. Available: http://doi.acm.org/10.1145/2494603.2480313

[53] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[54] I. Degtyarenko, O. Radyvonenko, K. Bokhan, and V. Khomenko, "Text/shape classifier for mobile applications with handwriting input," *Int. J. Doc. Anal.*

Recognit., vol. 19, no. 4, pp. 369–379, Dec. 2016. [Online]. Available: https://doi.org/10.1007/s10032-016-0276-0

[55] U. Dekel, "Supporting distributed software design meetings: What can we learn from co-located meetings?" in *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering*, ser. HSSE '05. New York, NY, USA: ACM, 2005, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/1082983.1083109

[56] U. Dekel and J. D. Herbsleb, "Notation and representation in collaborative object-oriented design: An observational study," in *Proceedings of the 22Nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications*, ser. OOPSLA '07. New York, NY, USA: ACM, 2007, pp. 261–280. [Online]. Available: http://doi.acm.org/10.1145/1297027.1297047

[57] A. Delaye and C.-L. Liu, *Text/Non-text Classification in Online Handwritten Documents with Conditional Random Fields*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 514–521. [Online]. Available: https://doi.org/10.1007/978-3-642-33506-8_63

[58] ——, "Context modeling for text/non-text separation in free-form online handwritten documents," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013, pp. 86 580C–86 580C.

[59] ——, "Contextual text/non-text stroke classification in online handwritten notes with conditional random fields," *Pattern Recognition*, vol. 47, no. 3, pp. 959 – 968, 2014, handwriting Recognition and other PR Applications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320313001878

[60] ——, "Multi-class segmentation of free-form online documents with tree conditional random fields," *Int. J. Doc. Anal. Recognit.*, vol. 17, no. 4, pp. 313–329, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10032-014-0221-z

[61] D. Deridder, "A concept-oriented approach to support software maintenance and reuse activities," in *In 5th Joint Conference on Knowledge-Based Software Engineering (JCKBSE). IOS Press - Series "Frontiers in Artificial Intelligence and Applications*. IOS Press, 2002.

[62] K. Devlin, *Sets, functions, and logic: an introduction to abstract mathematics*. CRC Press, 2003.

[63] DIN, *Schreib- und Gestaltungsregeln für die Textverarbeitung: Sonderdruck von DIN 5008:2011*, DIN e. V., Ed. Beuth, 2011.

[64] T. Dray and C. A. Manogue, "The geometry of the dot and cross products," *Journal of Online Mathematics and its Applications*, vol. 6, pp. 1–13, 2006.

[65] Dudenredaktion, *Duden - Die deutsche Rechtschreibung: Das umfassende Standardwerk auf der Grundlage der amtlichen Regeln*. Duden, 2017.

[66] S. T. Dumais, "Latent semantic analysis," *Annual review of information science and technology*, vol. 38, no. 1, pp. 188–230, 2004.

[67] J. Espinosa, S. Slaughter, R. Kraut, and J. Herbsleb, "Team knowledge and coordination in geographically distributed software development," *J. Manage. Inf. Syst.*, vol. 24, no. 1, pp. 135–169, Jul. 2007. [Online]. Available: http://dx.doi.org/10.2753/MIS0742-1222240104

[68] A. Evans, S. Kent, and B. Selic, *UML 2000 - The unified modeling language: Advancing the standard*. Springer, 2008.

[69] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, 2016.

[70] E. S. Ferguson, *Engineering and the Mind's Eye*. MIT Press, 1994.

[71] M. Fischer, *Website Boosting 2.0: Suchmaschinen-Optimierung, Usability, Online-Marketing*. mitp Verlags GmbH & Co. KG, 2009.

[72] E. Fleisch, M. Dierkes, and M. Kickuth, "Ubiquitous computing: Auswirkungen auf die industrie," *Industrie Management*, vol. 6, pp. 29–31, 2003.

[73] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics - Principles and Practice*. Addison Wesley, 1990.

[74] M. J. Fonseca, C. Pimentel, and J. A. Jorge, "Cali: An online scribble recognizer for calligraphic interfaces," in *AAAI spring symposium on sketch understanding*, 2002, pp. 51–58.

[75] C. Fox, "A stop list for general text," in *Acm sigir forum*, vol. 24, no. 1-2. ACM, 1989, pp. 19–21.

[76] W. B. Frakes and R. Baeza-Yates, Eds., *Information Retrieval: Data Structures and Algorithms*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[77] I. Galvao and A. Goknil, "Survey of traceability approaches in model-driven engineering," in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, ser. EDOC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 313–.

[78] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software*. Prentice Hall, 1994.

[79] R. Ganhör and W. Spreicer, "Monox: Extensible gesture notation for mobile devices," in *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services*, ser. MobileHCI '14. New York, NY, USA: ACM, 2014, pp. 203–212. [Online]. Available: http://doi.acm.org/10.1145/2628363.2628394

[80] F. Geyer, J. Budzinski, and H. Reiterer, "Ideavis: A hybrid workspace and interactive visualization for paper-based collaborative sketching sessions," in *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, ser. NordiCHI '12. New York, NY, USA: ACM, 2012, pp. 331–340. [Online]. Available: http://doi.acm.org/10.1145/2399016.2399069

[81] V. Goel, *Sketches of Thought*. MIT Press, 1995.

[82] G. Goldschmidt, "On visual design thinking: the vis kids of architecture," *Design Studies*, vol. 15, no. 2, pp. 158 – 174, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0142694X94900221

[83] Götze, Marco and Geyer, Steffen, "German Stopwords," 2018, [accessed 2018-02-09]. [Online]. Available: https://github.com/solariz/german_stopwords

[84] S. Grapenthin, M. Book, T. Richter, and V. Gruhn, "Supporting feature estimation with risk and effort annotations," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug 2016, pp. 17–24.

[85] S. Grapenthin, M. Book, V. Gruhn, C. Schneider, and K. Völker, "Reducing complexity using an interaction room: An experience report," in *Proceedings of the 31st ACM International Conference on Design of Communication*, ser. SIGDOC '13. New York, NY, USA: ACM, 2013, pp. 71–76. [Online]. Available: http://doi.acm.org/10.1145/2507065.2507087

[86] M. D. Gross and E. Y.-L. Do, "Ambiguous intentions: A paper-like interface for creative design," in *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '96. New York, NY, USA: ACM, 1996, pp. 183–192. [Online]. Available: http://doi.acm.org/10.1145/237091.237119

[87] A. Grosskopf, G. Decker, and M. Weske, *The process: business process modeling using BPMN*. Meghan Kiffer Press, 2009.

[88] J. Grudin and E. S. Poole, "Wikis at work: Success factors and challenges for sustainability of enterprise wikis," in *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, ser. WikiSym '10. New York, NY, USA: ACM, 2010, pp. 5:1–5:8. [Online]. Available: http://doi.acm.org/10.1145/1832772.1832780

[89] J. Grundy and J. Hosking, "Supporting generic sketching-based input of diagrams in a domain-specific visual language meta-tool," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 282–291. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.81

[90] F. Guimbretière, M. Stone, and T. Winograd, "Fluid interaction with high-resolution wall-size displays," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '01. New York, NY, USA: ACM, 2001, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/502348.502353

[91] R. Guindon, "Designing the design process: Exploiting opportunistic thoughts," *Hum.-Comput. Interact.*, vol. 5, no. 2, pp. 305–344, Jun. 1990. [Online]. Available: http://dx.doi.org/10.1207/s15327051hci0502&3_6

[92] R. Gumienny, L. Gericke, M. Wenzel, and C. Meinel, "Supporting creative collaboration in globally distributed companies," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, ser. CSCW '13. New York, NY, USA: ACM, 2013, pp. 995–1007. [Online]. Available: http://doi.acm.org/10.1145/2441776.2441890

[93] B. Hailpern and P. Tarr, "Model-driven development: The good, the bad, and the ugly," *IBM Syst. J.*, vol. 45, no. 3, pp. 451–461, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1147/sj.453.0451

[94] J. Hailpern, E. Hinterbichler, C. Leppert, D. Cook, and B. P. Bailey, "Team storm: Demonstrating an interaction model for working with multiple ideas during creative group work," in *Proceedings of the 6th ACM SIGCHI Conference on Creativity &Amp; Cognition*, ser. C&C '07. New York, NY, USA: ACM, 2007, pp. 193–202. [Online]. Available: http://doi.acm.org/10.1145/1254960.1254987

[95] M. Haller, J. Leitner, T. Seifried, J. R. Wallace, S. D. Scott, C. Richter, P. Brandl, A. Gokcezade, and S. Hunter, "The nice discussion room: Integrating paper and digital media to support co-located group meetings,"

in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 609–618. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753418

[96] T. Hammond and R. Davis, "Ladder: A language to describe drawing, display, and editing in sketch recognition," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003, pp. 461–467.

[97] ——, "Tahuti: a geometrical sketch recognition system for uml class diagrams," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1185657.1185786

[98] J. Hardy, C. Bull, G. Kotonya, and J. Whittle, "Digitally annexing desk space for software development (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 812–815. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985910

[99] M. Hazewinkel, Ed., *Encyclopaedia Of Mathematics*. Springer, 2010, vol. 8.

[100] C. Heide Damm, K. Marius Hansen, M. Thomsen, and M. Tyrsted, *Supporting Several Levels of Restriction in the UML*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 396–409. [Online]. Available: http://dx.doi.org/10.1007/3-540-40011-7_29

[101] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 188–198. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.11

[102] M. Hesenius, M. Kleffmann, and V. Gruhn, "Sketching gesture-based applications in a collaborative working environment with wall-sized displays," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, Oct 2016, pp. 327–336.

[103] M. Hesenius and V. Gruhn, "Introducing gesturecards: A prototyping gesture notation," in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, ser. NordiCHI '16. New York, NY, USA: ACM, 2016, pp. 111:1–111:6. [Online]. Available: http://doi.acm.org/10.1145/2971485.2996746

[104] M. Hesenius, S. Sternal, and V. Gruhn, "A multi-touch-recognizer for gesturecards," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '17. New York, NY, USA: ACM, 2017, pp. 75–80. [Online]. Available: http://doi.acm.org/10.1145/3102113.3102132

[105] O. Hilliges, L. Terrenghi, S. Boring, D. Kim, H. Richter, and A. Butz, "Designing for collaborative creative problem solving," in *Proceedings of the 6th ACM SIGCHI Conference on Creativity &Amp; Cognition*, ser. C&C '07. New York, NY, USA: ACM, 2007, pp. 137–146. [Online]. Available: http://doi.acm.org/10.1145/1254960.1254980

[106] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[107] A. Hosseini-Khayat, T. Seyed, C. Burns, and F. Maurer, "Low-fidelity prototyping of gesture-based applications," in *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '11. New York, NY, USA: ACM, 2011, pp. 289–294. [Online]. Available: http://doi.acm.org/10.1145/1996461.1996538

[108] L. Hoste and B. Signer, "Criteria, Challenges and Opportunities for Gesture Programming Languages," in *Proceedings of the 1st Workshop on Engineering Gestures for Multimodal Interfaces (EGMI 2014), co-located with the 6th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2014)*. CEUR Workshop Proceedings 1190, June 2014, pp. 22–29.

[109] C. D. Hundhausen, "Using end-user visualization environments to mediate conversations: A 'communicative dimensions' framework," *J. Vis. Lang. Comput.*, vol. 16, no. 3, pp. 153–185, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.jvlc.2004.11.002

[110] B. C. Hungerford, A. R. Hevner, and R. W. Collins, "Reviewing software diagrams: A cognitive study," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 82–96, Feb. 2004. [Online]. Available: http://dx.doi.org/10.1109/TSE.2004.1265814

[111] E. Indermühle, V. Frinken, and H. Bunke, "Mode detection in online handwritten documents using blstm neural networks," in *2012 International Conference on Frontiers in Handwriting Recognition*, Sept 2012, pp. 302–307.

[112] E. Indermühle, M. Liwicki, and H. Bunke, "Iamondo-database: An online handwritten document database with non-uniform contents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, ser. DAS '10. New York, NY, USA: ACM, 2010, pp. 97–104. [Online]. Available: http://doi.acm.org/10.1145/1815330.1815343

[113] S. Izadi, H. Brignull, T. Rodden, Y. Rogers, and M. Underwood, "Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*. ACM, 2003, pp. 159–168.

[114] G. Johnson, M. D. Gross, J. Hong, and E. Yi-Luen Do, "Computational support for sketching in design: A review," *Found. Trends Hum.-Comput. Interact.*, vol. 2, no. 1, pp. 1–93, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1561/1100000013

[115] A. Jones, A. Kendira, D. Lenne, T. Gidel, and C. Moulin, "The tatin-pic project: A multi-modal collaborative work environment for preliminary design," in *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*. IEEE, 2011, pp. 154–161.

[116] W. Ju, B. A. Lee, and S. R. Klemmer, "Range: Exploring implicit interaction through electronic whiteboard design," in *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '08. New York, NY, USA: ACM, 2008, pp. 17–26. [Online]. Available: http://doi.acm.org/10.1145/1460563.1460569

[117] D. Kammer, D. Henkens, C. Henzen, and R. Groh, "Gesture formalization for multitouch," *Software: Practice and Experience*, vol. 45, no. 4, pp. 527–548, 2015.

[118] S. H. Khandkar and F. Maurer, "A domain specific language to define gestures for multi-touch applications," in *Proceedings of the 10th Workshop on Domain-Specific Modeling*, ser. DSM '10. New York, NY, USA: ACM, 2010, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/2060329.2060339

[119] J.-W. Kim and T.-J. Nam, "EventHurdle: Supporting Designers' Exploratory Interaction Prototyping with Gesture-based Sensors," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. New York, NY, USA: ACM, 2013, pp. 267–276.

[120] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala, "Proton: Multitouch gestures as regular expressions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2885–2894. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208694

[121] M. Kleffmann, M. Book, and V. Gruhn, "Towards recovering and maintaining trace links for model sketches across interactive displays," in *Traceability in Emerging Forms of Software Engineering (TEFSE), 2013 International Workshop on*, May 2013, pp. 23–29.

[122] ——, "Navigation among model sketches on large interactive displays," in *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International*, Sept 2014, pp. 191–200.

[123] M. Kleffmann, S. Röhl, V. Gruhn, and M. Book, "Establishing and navigating trace links between elements of informal diagram sketches," in *Software and Systems Traceability (SST), 2015 IEEE/ACM 8th International Symposium on*, May 2015, pp. 1–7.

[124] M. Kleffmann, "Augir: The conceptual design and evaluation of an augmented interaction room," in *Proceedings of the 29th ACM/IEEE*

International Conference on Automated Software Engineering, ser. ASE '14. New York, NY, USA: ACM, 2014, pp. 883–886. [Online]. Available: http://doi.acm.org/10.1145/2642937.2653467

[125] M. Kleffmann, M. Book, and V. Gruhn, "Supporting collaboration of heterogeneous teams in an augmented team room," in *Proceedings of the 6th International Workshop on Social Software Engineering*, ser. SSE 2014. New York, NY, USA: ACM, 2014, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/2661685.2661688

[126] M. Kleffmann, M. Book, E. Hebisch, and V. Gruhn, "Automated versioning and temporal navigation for model sketches on large interactive displays," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 161–168. [Online]. Available: http://doi.acm.org/10.1145/2554850.2563668

[127] M. Kleffmann, M. Hesenius, and V. Gruhn, "Connecting ui and business processes in a collaborative sketching environment," in *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '15. New York, NY, USA: ACM, 2015, pp. 200–205. [Online]. Available: http://doi.acm.org/10.1145/2774225.2775076

[128] M. Kleffmann, S. Röhl, M. Book, and V. Gruhn, "Evaluation of a traceability approach for informal freehand sketches," *Automated Software Engineering*, pp. 1–43, 2017.

[129] S. R. Klemmer, M. W. Newman, R. Farrell, M. Bilezikjian, and J. A. Landay, "The designers' outpost: A tangible interface for collaborative web site," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '01. New York, NY, USA: ACM, 2001, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/502348.502350

[130] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, "An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks," *IEEE Trans. Softw. Eng.*, vol. 32, no. 12, pp. 971–987, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TSE.2006.116

[131] H. Koike, S. Nagashima, Y. Nakanishi, and Y. Sato, "Enhancedtable: An augmented table system for supporting face-to-face meeting in ubiquitous environment," in *Proceedings of the Second International Conference on Ubiquitous Computing Systems*, ser. UCS'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 117–130. [Online]. Available: http://dx.doi.org/10.1007/11526858_10

[132] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, pp. 69–81, Mar. 1995.

[133] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß, and M. Steinbrecher, "Computational intelligence: Eine methodische einführung in künstliche neuronale netze," *Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze Vieweg+ Teubner, Wiesbaden*, 2011.

[134] D. Kuhn, "Selecting and effectively using a computer aided software engineering tool," Westinghouse Savannah River Co., Aiken, SC (USA), Tech. Rep., 1989.

[135] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655813

[136] J. A. Landay, "Silk: Sketching interfaces like krazy," in *Conference Companion on Human Factors in Computing Systems*, ser. CHI '96. New York, NY, USA: ACM, 1996, pp. 398–399. [Online]. Available: http://doi.acm.org/10.1145/257089.257396

[137] J. A. Landay and B. A. Myers, "Interactive sketching for the early stages of user interface design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '95. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 43–50. [Online]. Available: http://dx.doi.org/10.1145/223904.223910

[138] ——, "Sketching interfaces: Toward more human interface design," *Computer*, vol. 34, no. 3, pp. 56–64, Mar. 2001. [Online]. Available: http://dx.doi.org/10.1109/2.910894

[139] J. H. Larkin and H. A. Simon, "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science*, vol. 11, no. 1, pp. 65 – 100, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0364021387800265

[140] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: A study of developer work habits," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 492–501. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134355

[141] P. Letelier, "A framework for requirements traceability in uml-based projects," in *In Proc. of 1st Intl. Workshop on Traceability in Emerging Forms of Softw. Eng*, 2002, pp. 32–41.

[142] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE Softw.*, vol. 20, no. 6, pp. 35–39, Nov. 2003. [Online]. Available: https://doi.org/10.1109/MS.2003.1241364

[143] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics-Doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[144] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.

[145] J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay, "Denim: finding a tighter fit between tools and practice for web site design," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ser. CHI '00. New York, NY, USA: ACM, 2000, pp. 510–517. [Online]. Available: http://doi.acm.org/10.1145/332040.332486

[146] J. Lin, M. Thomsen, and J. A. Landay, "A visual language for sketching large and complex interactive designs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '02. New York, NY, USA: ACM, 2002, pp. 307–314. [Online]. Available: http://doi.acm.org/10.1145/503376.503431

[147] N. Lindgren, "Machine recognition of human language part iii - cursive script recognition," *IEEE Spectrum*, vol. 2, no. 5, pp. 104–116, May 1965.

[148] U. Lipp, I. Sachsenmeier, H. Will, and B. Weidenmann, *Workshops, Seminare und Besprechungen: mit Kreativität und Methode zum sicheren Erfolg*, ser. Mit Kommunikation zum Erfolg. Beltz, 2009. [Online]. Available: https://books.google.de/books?id=WENiPpEEnwAC

[149] U. Lipp and H. Will, *Das große Workshop-Buch: Konzeption, Inszenierung und Moderation von Klausuren, Besprechungen und Seminaren.* Beltz, 2008.

[150] D. Loksa, N. Mangano, T. D. LaToza, and A. v. d. Hoek, "Enabling a classroom design studio with a collaborative sketch design tool," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 1073–1082. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486935

[151] H. Lü, J. A. Fogarty, and Y. Li, "Gesture script: Recognizing gestures and their structure using rendering scripts and interactively trained parts," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM, 2014, pp. 1685–1694.

[152] K. Machii, H. Fukushima, and M. Nakagawa, "On-line text/drawings segmentation of handwritten patterns," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, Oct 1993, pp. 710–713.

[153] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek, "How software designers interact with sketches at the whiteboard," *IEEE Transactions on Software Engineering*, vol. 41, no. 2, pp. 135–156, Feb 2015.

[154] N. Mangano, A. Baker, M. Dempsey, E. Navarro, and A. van der Hoek, "Software design sketching with calico," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 23–32. [Online]. Available: http://doi.acm.org/10.1145/1858996.1859003

[155] N. Mangano, A. Baker, and A. van der Hoek, "Calico: a prototype sketching tool for modeling in early design," in *Proceedings of the 2008 international workshop on Models in software engineering*, ser. MiSE '08. New York, NY, USA: ACM, 2008, pp. 63–68. [Online]. Available: http://doi.acm.org/10.1145/1370731.1370747

[156] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek, "Supporting informal design with interactive whiteboards," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 331–340. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557411

[157] N. Mangano and A. van der Hoek, "The design and evaluation of a tool to support software designers at the whiteboard," *Automated Software Engineering*, vol. 19, no. 4, pp. 381–421, 2012.

[158] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 03 1947. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177730491

[159] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 125–135.

[160] A. Marcus, J. I. Maletic, and A. Sergeyev, "Recovery of traceability links between software documentation and source code," *International Journal of Software Engineering and Knowledge Engineering*, vol. 15, pp. 811–836, 2005.

[161] A. Marcus, X. Xie, and D. Poshyvanyk, "When and how to visualize traceability links?" in *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, ser. TEFSE '05. New York, NY, USA: ACM, 2005, pp. 56–61.

[162] A. Marie Vans, A. von Mayrhauser, and G. Somlo, "Program understanding behavior during corrective maintenance of large-scale software," *Int. J. Hum.-Comput. Stud.*, vol. 51, no. 1, pp. 31–70, Jul. 1999. [Online]. Available: http://dx.doi.org/10.1006/ijhc.1999.0268

[163] G. Mark, "Extreme collaboration," *Commun. ACM*, vol. 45, no. 6, pp. 89–93, Jun. 2002. [Online]. Available: http://doi.acm.org/10.1145/508448.508453

[164] T. Memmel and H. Reiterer, "Inspector - interactive ui specification tool," in *In Proc. of the 7th International Conference On Computer Aided Design of User Interfaces (CADUI) 2008*. Springer, 2008, pp. 161–174.

[165] T. Mens and T. Tourwé, "A survey of software refactoring," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 126–139, Feb. 2004. [Online]. Available: http://dx.doi.org/10.1109/TSE.2004.1265817

[166] P. Mermelstein and M. Eyden, "A system for automatic recognition of handwritten words," in *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, ser. AFIPS '64 (Fall, part I). New York, NY, USA: ACM, 1964, pp. 333–342. [Online]. Available: http://doi.acm.org/10.1145/1464052.1464081

[167] Merriam-Webster, *Merriam-Webster's manual for writers and editors*. Merriam-Webster, 1998.

[168] Microsoft, "Microsoft.Ink," 2018, [accessed 2018-02-28]. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms826516.aspx

[169] ——, "Windows Presentation Foundation," 2018, [accessed 2018-02-28]. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wpf/index

[170] A. Mizera, E. Czeizler, and I. Petre, "Self-assembly models of variable resolution," in *Transactions on Computational Systems Biology XIV*, C. Priami, I. Petre, and E. Vink, Eds. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 181–203. [Online]. Available: http://dl.acm.org/citation.cfm?id=2434969.2434978

[171] K. Mochida and M. Nakagawa, "Separating drawings, formula and text from free handwriting," in *Proceedings of the 11th Conference of the International Graphonomics Society (IGS2003)*, 2003, pp. 216–219.

[172] ——, "Separating figures, mathematical formulas and japanese text from free handwriting in mixed online documents," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 07, pp. 1173–1187, 2004.

[173] L. G. Murta, A. Hoek, and C. M. Werner, "Continuous and automated evolution of architecture-to-implementation traceability links," *Automated Software Engg.*, vol. 15, no. 1, pp. 75–107, Mar. 2008.

[174] B. Myers, S. Y. Park, Y. Nakano, G. Mueller, and A. Ko, "How designers design and program interactive behaviors," in *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, Sept 2008, pp. 177–184.

[175] D. Mynatt, T. Igarashi, W. K. Edwards, and A. LaMarca, "Designing an augmented writing surface," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp. 55–61, 2000.

[176] E. D. Mynatt, T. Igarashi, W. K. Edwards, and A. LaMarca, "Flatland: New dimensions in office whiteboards," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '99. New York, NY, USA: ACM, 1999, pp. 346–353. [Online]. Available: http://doi.acm.org/10.1145/302979.303108

[177] M. W. Newman, J. Lin, J. I. Hong, and J. A. Landay, "Denim: An informal web site design tool inspired by observations of practice," *Hum.-Comput.*

Interact., vol. 18, no. 3, pp. 259–324, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI1803_3

[178] Z. Obrenovic and J.-B. Martens, "Sketching interactive systems with sketchify," *ACM Trans. Comput.-Hum. Interact.*, vol. 18, no. 1, pp. 4:1–4:38, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1959022.1959026

[179] J. S. Olson and S. Teasley, "Groupware in the wild: Lessons learned from a year of virtual collocation," in *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '96. New York, NY, USA: ACM, 1996, pp. 419–427. [Online]. Available: http://doi.acm.org/10.1145/240080.240353

[180] H. Ossher, R. Bellamy, B. John, and M. Desmond, "Concern development in software design discussions," in *Software Designers in Action*, ser. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. Chapman and Hall/CRC, Sep. 2013, pp. 293–312. [Online]. Available: http://dx.doi.org/10.1201/b15530-21

[181] S. Otte, D. Krechel, M. Liwicki, and A. Dengel, "Local feature based online mode detection with recurrent neural networks," in *2012 International Conference on Frontiers in Handwriting Recognition*, Sept 2012, pp. 533–537.

[182] R. Patel, B. Plimmer, J. Grundy, and R. Ihaka, "Ink features for diagram recognition," in *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, ser. SBIM '07. New York, NY, USA: ACM, 2007, pp. 131–138. [Online]. Available: http://doi.acm.org/10.1145/1384429.1384457

[183] B. Paulson and T. Hammond, "Paleosketch: Accurate primitive sketch recognition and beautification," in *Proceedings of the 13th International Conference on Intelligent User Interfaces*, ser. IUI '08. New York, NY, USA: ACM, 2008, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1378773.1378775

[184] E. R. Pedersen, K. McCall, T. P. Moran, and F. G. Halasz, "Tivoli: An electronic whiteboard for informal workgroup meetings," in *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. ACM, 1993, pp. 391–398.

[185] M. Petre, "Insights from expert software design practice," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 233–242. [Online]. Available: http://doi.acm.org/10.1145/1595696.1595731

[186] ——, "Uml in practice," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 722–731. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486883

[187] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/34.824821

[188] B. Plimmer and M. Apperley, "Interacting with sketched interface designs: an evaluation study," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '04. New York, NY, USA: ACM, 2004, pp. 1337–1340. [Online]. Available: http://doi.acm.org/10.1145/985921.986058

[189] B. Plimmer and J. Grundy, "Beautifying sketching-based design tool content: Issues and experiences," in *Proceedings of the Sixth Australasian Conference on User Interface - Volume 40*, ser. AUIC '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 31–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=1082243.1082248

[190] D. Pye, *The Nature and Aesthetics of Design*. A&C Black, 2000.

[191] S. Qin, "Intelligent classification of sketch strokes," in *EUROCON 2005 - The International Conference on "Computer as a Tool"*, vol. 2, Nov 2005, pp. 1374–1377.

[192] M. Renger, G. L. Kolfschoten, and G.-J. Vreede, "Using interactive whiteboard technology to support collaborative modeling," in *Groupware: Design, Implementation, and Use*, R. O. Briggs, P. Antunes, G.-J. Vreede, and A. S. Read, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 356–363. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-92831-7_29

[193] M. Rettig, "Prototyping for tiny fingers," *Commun. ACM*, vol. 37, no. 4, pp. 21–27, Apr. 1994. [Online]. Available: http://doi.acm.org/10.1145/175276.175288

[194] J. T. Richardson, "Eta squared and partial eta squared as measures of effect size in educational research," *Educational Research Review*, vol. 6, no. 2, pp. 135 – 147, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1747938X11000029

[195] J. A. Rodríguez, G. Sánchez, and J. Lladós, "Categorization of digital ink elements using spectral features," in *International Workshop on Graphics Recognition*. Springer, 2007, pp. 181–190.

[196] D. Rubine, "Specifying gestures by example," in *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '91. New York, NY, USA: ACM, 1991, pp. 329–337. [Online]. Available: http://doi.acm.org/10.1145/122718.122753

[197] I. Ruedel, *Workshops: optimal vorbereiten, spannend inszenieren, professionell nachbereiten*. Linde Verlag GmbH, 2008.

[198] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[199] U. Sangiorgi and J. Vanderdonckt, "Gambit: Addressing multi-platform collaborative sketching with html5," in *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '12. New York, NY, USA: ACM, 2012, pp. 257–262. [Online]. Available: http://doi.acm.org/10.1145/2305484.2305527

[200] P. Sapna and H. Mohanty, "Ensuring consistency in relational repository of uml models," in *Information Technology, (ICIT 2007). 10th International Conference on*, dec. 2007, pp. 217 –222.

[201] H. F. Schantz, *History of OCR, Optical Character Recognition.* Recognition Technologies Users Association, 1982.

[202] W. Scholl and R. Drews, *Handbuch Mathematik.* Orbis Verlag, 2001.

[203] J. Schumann, T. Strothotte, S. Laser, and A. Raab, "Assessing the effect of non-photorealistic rendered images in cad," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '96. New York, NY, USA: ACM, 1996, pp. 35–41. [Online]. Available: http://doi.acm.org/10.1145/238386.238398

[204] M. Schütze, P. Sachse, and A. Römer, "Support value of sketching in the design process," *Research in Engineering Design*, vol. 14, no. 2, pp. 89–97, 2003. [Online]. Available: http://dx.doi.org/10.1007/s00163-002-0028-7

[205] K. Schwaber, *Agile project management with Scrum.* Microsoft press, 2004.

[206] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Softw.*, vol. 20, no. 5, pp. 42–45, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1109/MS.2003.1231150

[207] R. Settimi, J. Cleland-Huang, O. Ben Khadra, J. Mody, W. Lukasik, and C. DePalma, "Supporting software evolution through dynamically retrieving traces to uml artifacts," in *Software Evolution, 2004. Proceedings. 7th International Workshop on Principles of*, sept. 2004, pp. 49 – 54.

[208] B. Sharif and H. Kagdi, "On the use of eye tracking in software traceability," in *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, ser. TEFSE '11. New York, NY, USA: ACM, 2011, pp. 67–70.

[209] C. Shen, K. Everitt, and K. Ryall, "Ubitable: Impromptu face-to-face collaboration on horizontal interactive surfaces," in *UbiComp 2003: Ubiquitous Computing*. Springer, 2003, pp. 281–288.

[210] M. Shilman and P. Viola, "Spatial recognition and grouping of text and graphics," in *Proceedings of the First Eurographics Conference on Sketch-Based Interfaces and Modeling*, ser. SBM'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 91–95. [Online]. Available: http://dx.doi.org/10.2312/SBM/SBM04/091-095

[211] F. M. Shipman and C. C. Marshall, "Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems," *Computer Supported Cooperative Work (CSCW)*, vol. 8, no. 4, pp. 333–352, 1999.

[212] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, "An examination of software engineering work practices," in *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research*, ser. CASCON '97. IBM Press, 1997, pp. 21–. [Online]. Available: http://dl.acm.org/citation.cfm?id=782010.782031

[213] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 24, no. 4, pp. 35–42, 2001. [Online]. Available: http://singhal.info/ieee2001.pdf

[214] SMART Technologies, "SMART Board," 2018, [accessed 2018-02-09]. [Online]. Available: https://home.smarttech.com/interactive-displays-for-business

[215] R. W. Soukoreff and I. S. MacKenzie, "Measuring errors in text entry tasks: An application of the levenshtein string distance statistic," in *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA

'01. New York, NY, USA: ACM, 2001, pp. 319–320. [Online]. Available: http://doi.acm.org/10.1145/634067.634256

[216] G. Spanoudakis, "Plausible and adaptive requirement traceability structures," in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, ser. SEKE '02. New York, NY, USA: ACM, 2002, pp. 135–142.

[217] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," in *in Handbook of Software Engineering and Knowledge Engineering*. World Scientific, 2001, pp. 329–380.

[218] G. Stapleton, B. Plimmer, A. Delaney, and P. Rodgers, "Combining sketching and traditional diagram editing tools," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 1, pp. 10:1–10:29, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2631925

[219] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar, "Wysiwis revised: Early experiences with multiuser interfaces," *ACM Trans. Inf. Syst.*, vol. 5, no. 2, pp. 147–167, Apr. 1987. [Online]. Available: http://doi.acm.org/10.1145/27636.28056

[220] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the chalkboard: Computer support for collaboration and problem solving in meetings," *Commun. ACM*, vol. 30, no. 1, pp. 32–47, Jan. 1987. [Online]. Available: http://doi.acm.org/10.1145/7885.7887

[221] N. Streitz, T. Prante, C. Müller-Tomfelde, P. Tandler, and C. Magerkurth, "Roomware: The second generation," in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '02. New York, NY, USA: ACM, 2002, pp. 506–507. [Online]. Available: http://doi.acm.org/10.1145/506443.506452

[222] N. A. Streitz, J. Geißler, J. M. Haake, and J. Hol, "Dolphin: Integrated meeting support across liveboards, local and remote desktop environments," 1994.

[223] N. A. Streitz, J. Geißler, and T. Holmer, "Roomware for cooperative buildings: Integrated design of architectural spaces and information spaces," in *International Workshop on Cooperative Buildings*. Springer, 1998, pp. 4–21.

[224] N. A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz, "i-land: an interactive landscape for creativity and innovation," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999, pp. 120–127.

[225] Student, "The probable error of mean," *Biometrika*, vol. 6, no. 1, pp. 1–25, 1908. [Online]. Available: http://dx.doi.org/10.1093/biomet/6.1.1

[226] M. Sugimoto, K. Hosoi, and H. Hashizume, "Caretta: A system for supporting face-to-face collaboration by integrating personal and shared spaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 41–48. [Online]. Available: http://doi.acm.org/10.1145/985692.985698

[227] I. E. Sutherland, "Sketch pad: A man-machine graphical communication system," in *Proceedings of the SHARE Design Automation Workshop*, ser. DAC '64. New York, NY, USA: ACM, 1964, pp. 6.329–6.346. [Online]. Available: http://doi.acm.org/10.1145/800265.810742

[228] M. Suwa, J. Gero, and T. Purcell, "Unexpected discoveries and s-invention of design requirements: important vehicles for a design process," *Design Studies*, vol. 21, no. 6, pp. 539 – 567, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142694X99000344

[229] M. Suwa and B. Tversky, "What do architects and students perceive in their design sketches? a protocol analysis," *Design Studies*, vol. 18, no. 4, pp. 385–402, 10 1997.

[230] P. Tandler, "Architecture of beach: The software infrastructure for roomware environments," in *CSCW 2000: Workshop on Shared Environments to Support Face-to-Face Collaboration*, 2000, pp. 2–6.

[231] S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson, "How does radical collocation help a team succeed?" in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, ser. CSCW '00. New York, NY, USA: ACM, 2000, pp. 339–346. [Online]. Available: http://doi.acm.org/10.1145/358916.359005

[232] The Unicode Consortium, *The Unicode Standard*. Unicode Consortium, 2017.

[233] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Int. Res.*, vol. 37, no. 1, pp. 141–188, Jan. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1861751.1861756

[234] B. Tversky, "What do sketches say about thinking?" in *Proceedings of AAAI spring symposium on sketch understanding*, 2002.

[235] B. Tversky, M. Suwa, M. Agrawala, J. Heiser, C. Stolte, P. Hanrahan, D. Phan, J. Klingner, M.-P. Daniel, P. Lee, and J. Haymaker, *Human Behaviour in Design: Individuals, Teams, Tools*. Springer, 2003, ch. Sketches for Design and Design of Sketches, pp. 79–86.

[236] D. G. Ullman, S. Wood, and D. Craig, "The importance of drawing in the mechanical design process," *Computers & Graphics*, vol. 14, no. 2, pp. 263 – 274, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/009784939090037X

[237] T. Van Phan and M. Nakagawa, "Text/non-text classification in online handwritten documents with recurrent neural networks," in *2014 14th International Conference on Frontiers in Handwriting Recognition*, Sept 2014, pp. 23–28.

[238] ——, "Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents," *Pattern Recogn.*, vol. 51, no. C, pp. 112–124, Mar. 2016. [Online]. Available: https://doi.org/10.1016/j.patcog.2015.07.012

[239] R. van Solingen, E. Berghout, and F. van Latum, "Interrupts: Just a minute never is," *IEEE Softw.*, vol. 15, no. 5, pp. 97–103, Sep. 1998. [Online]. Available: http://dx.doi.org/10.1109/52.714843

[240] R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, "Gestures as point clouds," in *Proceedings of the 14th ACM international conference on Multimodal interaction - ICMI '12*. New York, New York, USA: ACM Press, Oct. 2012, p. 273.

[241] G. Venolia, "Textual allusions to artifacts in software-related repositories," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, ser. MSR '06. New York, NY, USA: ACM, 2006, pp. 151–154. [Online]. Available: http://doi.acm.org/10.1145/1137983.1138018

[242] P. Viriyakattiyaporn and G. C. Murphy, "Improving program navigation with an active help system," in *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, ser. CASCON '10. Riverton, NJ, USA: IBM Corp., 2010, pp. 27–41. [Online]. Available: http://dx.doi.org/10.1145/1923947.1923951

[243] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974. [Online]. Available: http://doi.acm.org/10.1145/321796.321811

[244] J. Walny, J. Haber, M. Dörk, J. Sillito, and S. Carpendale, "Follow that sketch: Lifecycles of diagrams and sketches in software development," in *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, Sept 2011, pp. 1–8.

[245] R. Waranusast, P. Haddawy, and M. Dailey, *Segmentation of Text and Non-text in On-Line Handwritten Patient Record Based on Spatio-Temporal Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 345–354. [Online]. Available: https://doi.org/10.1007/978-3-642-02976-9_47

[246] R. L. Wasserstein and N. A. Lazar, "The asa's statement on p-values: Context, process, and purpose," *The American Statistician*, vol. 70, no. 2, pp. 129–133, 2016. [Online]. Available: http://dx.doi.org/10.1080/00031305.2016.1154108

[247] M. Weber, M. Liwicki, Y. T. Schelske, C. Schoelzel, F. Strauß, and A. Dengel, "Mcs for online mode detection: Evaluation on pen-enabled multi-touch interfaces," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 957–961.

[248] D. Willems, S. Rossignol, and L. Vuurpijl, "Mode detection in on-line pen drawing and handwriting recognition," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ser. ICDAR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 31–35. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2005.160

[249] L. A. Williams and R. R. Kessler, "All i really need to know about pair programming i learned in kindergarten," *Commun. ACM*, vol. 43, no. 5, pp. 108–114, May 2000. [Online]. Available: http://doi.acm.org/10.1145/332833.332848

[250] S. Winkler and J. Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software and Systems Modeling (SoSyM)*, vol. 9, no. 4, pp. 529–565, 2010.

[251] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[252] J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1083–1092. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518866

[253] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes," in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '07. New York, NY, USA: ACM, 2007, pp. 159–168.

[254] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

[255] Y. Y. Wong, "Rough and ready prototypes: Lessons from graphic design," in *Posters and Short Talks of the 1992 SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '92. New York, NY, USA: ACM, 1992, pp. 83–84. [Online]. Available: http://doi.acm.org/10.1145/1125021.1125094

[256] ——, "Rough and ready prototypes: Lessons from graphic design," in *Posters and Short Talks of the 1992 SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '92. New York, NY, USA: ACM, 1992, pp. 83–84. [Online]. Available: http://doi.acm.org/10.1145/1125021.1125094

[257] D. Wüest, N. Seyff, and M. Glinz, "Flexisketch: A mobile sketching tool for software modeling," in *Mobile Computing, Applications, and Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, D. Uhler, K. Mehta, and J. Wong, Eds. Springer Berlin Heidelberg, 2013, vol. 110, pp. 225–244.

[258] D. Wüest, N. Seyff, and M. Glinz, "Flexisketch team: Collaborative sketching and notation creation on the fly," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 685–688. [Online]. Available: http://dl.acm.org/citation.cfm?id=2819009.2819138

[259] K. Yatani, E. Chung, C. Jensen, and K. N. Truong, "Understanding how and why open source contributors use diagrams in the development of ubuntu," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 995–1004. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518853

[260] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers," *Information and Software Technology*, vol. 49, no. 6, pp. 637 – 653,

2007, qualitative Software Engineering Research. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584907000122

[261] C. Zannier and F. Maurer, "Comparing decision making in agile and non-agile software organizations," in *Proceedings of the 8th International Conference on Agile Processes in Software Engineering and Extreme Programming*, ser. XP'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768961.1768963

[262] X. D. Zhou and C. L. Liu, "Text/non-text ink stroke classification in japanese handwriting based on markov random fields," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 1, Sept 2007, pp. 377–381.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt durch meine Unterschrift, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, habe ich als solche kenntlich gemacht. Die Arbeit lag in dieser oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vor.

Markus Kleffmann

Solingen, 28.03.2018