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

Entwurf eines Requirements Engineering Workflows für erklärbare Systeme

Masterarbeit

im Studiengang Informatik

von

Merve Balci

Prüfer: Prof. Dr. rer. nat. Kurt Schneider
Zweitprüfer: Dr. rer. nat. Jil Ann-Christin Klünder
Betreuer: Larissa Chazette

Hannover, 03. November 2021

Erklärung der Selbstständigkeit

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

Hannover, den 03. November 2021

Merve Balci

Merve Balci

Zusammenfassung

Nachdem die Forschung zur Softwareanforderung *Erklärbarkeit* sich hauptsächlich auf das Fachgebiet der künstlichen Intelligenz beschränkt hat, wird dieser aktuell aus der endnutzerzentrierten Perspektive betrachtet. Das Requirements Engineering (RE) sowie die Mensch-Maschine Kommunikation (MMK) befassen sich mit der Erfassung der Bedürfnisse, Ziele und des Kontextes um Erklärbarkeitsanforderungen zu definieren und Evaluationsmöglichkeiten bereitzustellen. Trotz dessen existiert noch keine Darstellung der Methoden und Aktivitäten die für die Entwicklung erklärbarer Systeme verwendet werden sollten. Im Rahmen dieser Masterarbeit werden Methoden aus dem RE und der MMK vorgestellt, die anhand eines Literaturreviews innerhalb wissenschaftlicher Arbeiten verwendet oder empfohlen wurden. Es werden Aktivitäten definiert, die den Phasen des RE zugeordnet werden, wodurch letztendlich ein Workflow für die Entwicklung erklärbarer Systeme entsteht, in der die Anforderunganalyse, das Anforderungsmanagement und Evaluationsmethodiken im Fokus stehen. Anschließend wird der Workflow durch 19 Experten aus der Industrie innerhalb von Experteninterviews evaluiert. Hierfür erstellten die Experten, basierend auf ihren Erfahrungen und ihrem Wissen aus der Industrie, einzeln ein Workflow für die Entwicklung erklärbarer Systeme. Die Ergebnisse werden mit dem auf der Literatur basierenden Workflows verglichen, um die Anwendbarkeit im Unternehmenskontext zu erfassen.

Abstract

The research on the software requirement *explainability* has been mainly limited to the field of artificial intelligence. Currently *explainability* is considered from the end-user centric perspective. Requirements engineering (RE), as well as human-computer interaction (HCI), are concerned with capturing the needs, goals, and context to define end-user centered explainability requirements and provide evaluation capabilities. Despite this, there is still no representation of the methods and activities that should be used and done for the development of explainable systems. In this master thesis methods from the RE and HCI are presented, which have been used or recommended within scientific papers based on a literature review. Activities are defined and assigned to the phases of RE, ultimately creating a workflow for the development of explainable systems in which requirements analysis, requirements management, and evaluation methodologies are the focus. The workflow is then evaluated by 19 experts from the industry within expert interviews. For this purpose, the experts, based on their experience and knowledge from the industry, individually created a workflow for the development of explainable systems. The results are compared with the literature-based workflows to capture the applicability in a corporate context.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Interesse an der Thematik der *Erklärbarkeit* ist in den letzten Jahren stark gestiegen. Das steigende Interesse ist motiviert durch die zunehmende Komplexität der künstlichen Intelligenz und der ebenso zunehmenden Integrierung von Softwaresystemen in allen Lebensbereichen. Der Einfluss von Softwaresystemen auf den Alltag lässt sich heutzutage in allen Lebenslagen beobachten. [87] Durch die Komplexität werden Softwaresysteme undurchsichtig, sodass Endnutzer und sogar Experten nicht mehr nachvollziehen können, wie die Software bedient werden soll und auf welchen Faktoren die Ergebnisse und Verhaltensweisen beruhen. Beim Onlineshopping schlägt sie uns Produkte vor oder berechnet die schnellste Route zum Ziel. Ergebnisse, Entscheidungen, Empfehlungen, Voraussagen oder Handlungen eines Softwaresystems beeinflussen unsere Handlungen und Entscheidungen in der Realität. Ebenso finden Softwaresysteme in kritischen Kontexten wie unter anderem dem Gesundheitswesen, dem Militär, dem Finanzsektor, dem Justizsystem oder innerhalb autonomer Fahrzeuge Anwendung. [33] Beispielsweise unterstützen sie Ärzte bei der Diagnose einer Krankheit. Durch die zunehmende Komplexität der zugrundeliegenden Algorithmen wird es gleichzeitig auch immer schwieriger nachzuvollziehen, auf welchen Faktoren die Vorschläge und Ergebnisse beruhen. So müssen Experten, wie Ärzte, auf Basis dieser undurchsichtigen Berechnungen weitreichende Entscheidungen treffen. [69] Die Erklärbarkeit befasst sich mit dieser Problematik von Softwaresystemen, weshalb sie in der Forschung derzeit große Aufmerksamkeit erfährt. [41] Politik, Ethik und Wissenschaft verlangt nach erklärbarer Software um die Ergebnisse und Algorithmen bewerten und nachvollziehen zu können. [12] 2016 wurde in der General Data Protection Regulation (GDPR) Endnutzern, die keine Experten auf dem Gebiet des maschinellen Lernens sind, das "Recht auf Erklärung" zugesprochen, wobei die Erklärungen kurz, transparent, verständlich und leicht zugänglich sein sollen. [12, 54] Im vereinigten Königreich argumentiert das *House of Lords AI Committee*, dass verständliche Systeme eine fundamentale Notwendigkeit

sind, wenn die Gesellschaft vertrauenswürdige Softwaresysteme verlangt. [94] Um den Aufruf nach erklärbaren Systemen nachzukommen, beschäftigt sich diese Masterarbeit mit dem Entwurf eines Workflows für die Entwicklung von erklärbaren Systemen.

1.1 Problemstellung

Obwohl der Aufruf nach erklärbaren Systemen groß ist, ist die Frage, wie erklärbare Systeme entwickelt werden können noch nicht geklärt. Damit das Recht für Erklärungen für den Endnutzer erfüllt werden kann, gilt es zunächst eine Antwort auf diese Frage zu finden. Folgende Forschungsfragen zielen auf die Lösung der Problematik ab:

Forschungsfrage 1: Welche Aktivitäten und Methoden eignen sich für die Entwicklung von erklärbaren Systemen in einem Softwareentwicklungsprozess?

Forschungsfrage 2: Wie kann ein Workflow für die Entwicklung von erklärbaren Systemen aussehen?

Forschungsfrage 3: Welche Einschränkungen und Probleme ergeben sich in der tatsächlichen Anwendung des Workflows in Unternehmen?

1.2 Lösungsansatz

Zur Beantwortung der Forschungsfragen wird in dieser Masterarbeit analysiert, wie das Requirements Engineering für die Entwicklung erklärbarer Systeme durchgeführt werden kann. Hierfür werden notwendige Aktivitäten definiert und geeignete Methoden aus den Fachgebieten der Mensch-Maschine Kommunikation (MMK) und des Requirement Engineerings (RE) ermittelt. Die MMK wird herangezogen, da erklärbare Systeme in einem menschzentrierten Entwicklungsprozess entwickelt werden müssen.

Es wird ein Mixed-Method Verfahren verwendet, in der zunächst ein Literaturreview und anschließend eine Interviewstudie durchgeführt wird. Mit dem Literaturreview werden in der Forschung verwendete oder empfohlene Methoden und Aktivitäten gesammelt. Anhand dieser Sammlung, sowie thematisch verwandten Arbeiten, wird ein Workflow entworfen, der auf der Literatur für Erklärbarkeit basiert um die Forschungsfrage 1 und 2 zu beantworten. Anschließend werden Experteninterviews durchgeführt, um das Wissen und die Erfahrung von Experten aus der Industrie zu erheben. Die Experten werden befragt welche Methoden und Aktivitäten sie aktuell verwenden und welche sie als geeignet bewerten, um erklärbare Systeme

im Unternehmenskontext zu entwickeln. Anhand dieser Ergebnisse wird ein finaler Workflow für die Entwicklung von erklärbaren Systemen aufgestellt und somit die Forschungsfragen (FF) 1, 2 und 3 beantwortet. Daraus ergeben sich im Rahmen dieser Masterarbeit folgende Phasen:

- 1. Phase – Durchführung eines Literaturreviews um in wissenschaftlichen Arbeiten verwendete Methoden und Aktivitäten für die Erstellung eines erklärbaren Systems herauszufinden. (FF1)
- 2. Phase – Entwurf eines Workflowmodels basierend auf den Ergebnissen von Phase 1. (FF2)
- 3. Phase – Evaluierung des Workflowmodels mit Experteninterviews. (FF1,2,3)
- 4. Phase – Überarbeitung des Workflowmodels basierend auf den Ergebnissen von Phase 3. (FF2,3)

1.3 Struktur der Arbeit

In dieser Masterarbeit wird ein phasenweises Vorgehen benötigt, weshalb sie wie folgt strukturiert wird. Zunächst wird das theoretische Wissen im Kapitel 2 zusammengefasst und erläutert. Kapitel 3 grenzt diese Arbeit von verwandten Arbeiten ab. Die folgenden Kapitel beruhen auf der Basis des Grundlagenwissens und der Erkenntnisse aus den verwandten Arbeiten. In Kapitel 4 wird detailliert auf die verwendeten methodischen Vorgehensweisen eingegangen und Entscheidungsvorgänge werden begründet. In Kapitel 5 werden die Ergebnisse aus dem Literaturreview vorgestellt und tabellarisch zusammengefasst. Anhand der Ergebnisse wird in begründeter Form die erste Version des Workflows zur Entwicklung von erklärbaren Systemen entworfen. Kapitel 6 thematisiert die Ergebnisse aus den Experteninterviews. Anhand der Ergebnisse wird der literaturbasierte Workflow bewertet und optimiert. Schlussendlich fasst Kapitel 7 die Ergebnisse zusammen und stellt einen Ausblick dar.

Kapitel 2

Grundlagen

Diese Masterarbeit verbindet das Requirements Engineering mit den Themengebieten der Mensch-Maschine Kommunikation und der Erklärbarkeit. In diesem Kapitel werden die drei großen Themengebiete erläutert, um das nötige Vorwissen für die folgenden Kapitel zu schaffen.

2.1 Requirements Engineering

Im Requirements Engineering werden systematisch Anforderungen analysiert und organisiert, was einen Teilaspekt innerhalb des Softwareentwicklungsprozesses darstellt. Das Ziel ist es relevante Anforderungen zu verstehen, mit den Stakeholdern[1] einen Konsens zu schaffen, Anforderungen zu dokumentieren und sie systematisch zu verwalten. Dafür werden die Wünsche und Bedürfnisse der Stakeholder analysiert und aufgezeichnet, wodurch das Risiko minimiert wird, diese nicht zu erfüllen. Die korrekte Erfassung und Dokumentation legt die Grundlage zur Erfüllung der Wünsche und der Bedürfnisse. Dies ist daher von großer Bedeutung für die erfolgreiche Entwicklung eines Softwaresystems. [89] Die Abbildungen 2.1 und 2.2 stellen die einzelnen Phasen aus den zwei Aktivitäten des Requirements Engineering nach Börger et al. [16] dar. Die Phasen werden mit den wichtigsten Methoden nach Paetsch et al. [82] unterlegt.[2]

2.1.1 Softwareanforderung

Softwareanforderungen geben die Bedürfnisse der Stakeholder bezüglich des Softwaresystems wieder. Sie werden unterschieden in *funktionale* und *nicht-*

[1] Ein Stakeholder ist jemand, der an einer Handlung (hier Softwaresystem) beteiligt oder von ihr betroffen ist. In dieser Arbeit wird der Begriff als ein Oberbegriff verwendet, aber gegebenenfalls wird auf spezifische Stakeholderklassen verwiesen (wie dem Endnutzer). [7]

[2] Die vorgeschlagenen Methoden wurden in der Literatur von Tjeerd et al. [91] explizit für die Entwicklung erklärbarer Systeme vorgeschlagen, weshalb die wissenschaftliche Arbeit von Paetsch et al. [82] für die Grundlagendarstellung herangezogen wurde.

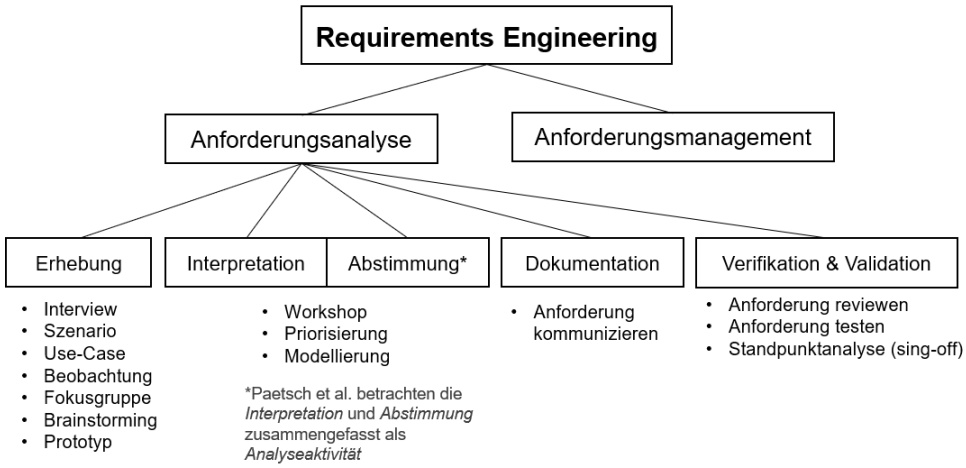

Abbildung 2.1: Darstellung der von Paetsch et al. [82] vorgestellten Methoden für die Anforderungsanalyse im Requirements Engineering. Die Struktur und Aufteilung der Abbildung ist entnommen von Börger et al. [16]

funktionale Anforderungen. Eine funktionale Anforderung beschreibt die Funktionalität eines Softwaresystems. Sie stellt dar *Was* das System in der Lage sein muss zu tun. [7]
Eine nicht-funktionale Anforderung beschreibt die gewünschte Qualitätseigenschaft des Softwaresystems. Sie stellt keine direkte Funktionalität dar, sondern beschreibt *Wie* das System funktionieren soll. Somit haben sie auch Einfluss auf die Art und Weise wie funktionale Anforderungen letztendlich umgesetzt werden. [7] Unvollständige oder fehlerhafte Anforderungen stellen die Hauptursache für das Scheitern von Softwareprojekten dar. Dadurch ist das Requirements Engineering ein essenzieller Bestandteil von Softwareentwicklungsprozessen. [72]

2.1.2 Anforderungsanalyse

Die Anforderungsanalyse nach Börger et al. [16] gliedert sich in folgende fünf Phasen auf.

Erhebung

In der Anforderungserhebung werden die Anforderungsquellen identifiziert. Anforderungsquellen sind unter anderem die Stakeholder und der Projektkontext. Beispielsweise können kontextspezifische Normen, Standards und Gesetze bestehen, die es bei der Softwareentwicklung einzuhalten gilt. Nach der Quellenidentifikation werden die Stakeholder befragt, der Projektkontext definiert und Softwareanforderungen ermittelt. Das Ziel ist es ein tiefgehendes Verständnis der Stakeholder, des Anwendungsbereichs,

der Systembeschränkungen, der Geschäftsanforderungen und des übergreifenden Problems zu entwickeln. Geeignete Methoden sind hierfür *Interviews, Szenarien, Use-Cases, Beobachtungen, Fokusgruppen, Brainstorming und Prototypen.* Diese Ergebnisse aus den Methoden werden zunächst in einer beliebigen Form wie Aufzeichnungen, Protokolle oder einfache Notizen dokumentiert. Aus der Anforderungserhebung soll die Produktvision und das Projektziel ersichtlich werden. [82]

Interpretation und Abstimmung

In der Interpretation werden die Informationen aus der Erfassungsphase konkretisiert, wodurch die Anforderungen definiert werden. Durch eine Konkretisierung der Anforderungen können diese auf ihre Notwendigkeit, Konsistenz, Vollständigkeit und Machbarkeit geprüft werden. In der Abstimmung sollen insbesondere Inkonsistenzen und Konflikte (Trade-offs) in den Anforderungen gefunden und gelöst werden. Dafür werden Priorisierungen der Anforderungen mit beteiligten Stakeholdern durchgeführt, um besonders wichtige Anforderungen zu identifizieren und Lösungen bzw. Kompromisse bei Problemen zu vereinbaren. Die wichtigsten Methoden hierfür sind *Workshops, Modellierungen der Anforderungen*[3] und die Anforderungspriorisierung. [82]

Dokumentation

In der Dokumentation werden die erhobenen Anforderungen in einer Anforderungsspezifikation festgehalten. Die Spezifikation beschreibt die geforderten und benötigten Anforderungen an die Software in einer ausführlichen Art, mit denen das Projektziel erreicht werden kann. Dabei ist das Ziel eine Kommunikation der Anforderungen mit allen beteiligten Stakeholdern und Entwicklern zu ermöglichen. Daher sollte eine Spezifikation eindeutig, vollständig, korrekt, verständlich, konsistent, prägnant und realisierbar sein. Sie bietet schlussendlich die Grundlage für die Bewertung des Softwaresystems und für die Änderungskontrolle. [82] [7]

Verifikation und Validation

In der Verifikation und Validation wird überprüft, ob die in der Dokumentation festgelegten Anforderungen angemessen erfüllt wurden. Dabei werden die Anwendungsziele im Verwendungskontext betrachtet. Sind einige Anforderungen nicht zufriedenstellend erfüllt worden, werden geeignete Maßnahmen für die Behebung der Probleme festgelegt. Geeignete Methoden sind hierfür *Reviews, Tests* und eine *Standpunktanalyse* durch Stakeholder, bei der das Softwaresystem abgelehnt (sing-off) oder angenommen wird. [82]

[3] Beispielsweise mit Datenfluss- oder objektorientierten Modellen. [82]

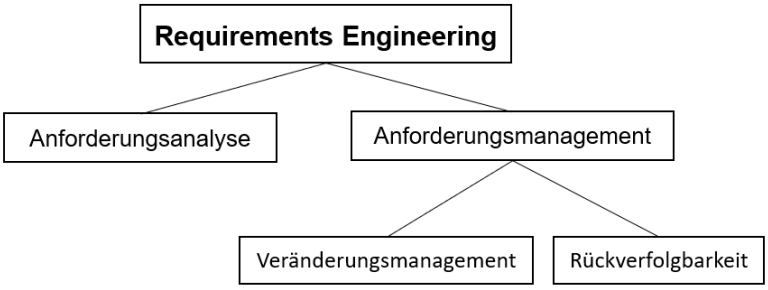

Abbildung 2.2: Darstellung des Anforderungsmanagements im Requirements Engineering nach Börger et al. [16] Hierfür wurden von Paetsch et al. [82] keine geeigneten Methoden genannt.

2.1.3 Anforderungsmanagement

Das Anforderungsmanagement gliedert sich in zwei Phasen auf, in denen Anforderungen verwaltet werden (siehe Abbildung 2.2). Das Ziel ist es Informationen zu erfassen, zu speichern, zu verbreiten und zu verwalten. [82]

Veränderungsmanagement

Das Veränderungsmanagement verwaltet alle Änderungen und Versionen der Anforderungen. Diese Änderungen und Anforderungsversionen werden festgehalten und an die Entwickler weitergegeben, sodass diese Nachvollzogen werden können. [7]

Rückverfolgbarkeit

Die Rückverfolgbarkeit beschäftigt sich mit der Nachverfolgung von Entscheidungen und dem Anforderungsstatus. Die Rückverfolgbarkeit der Anforderungen ermöglicht es sie in eine Beziehung zu der Umsetzung (Design und Implementierung des Systems) und anderen Anforderungen zu stellen. Dabei werden Verknüpfungen zwischen Annahmen und Quellen der Anforderungen (pre-tracing), sowie die Auswirkungen von Anforderungen im System (post-tracing) unterschieden. [82] [7]

2.2 Erklärbarkeit

Die Erklärbarkeit ist eine nicht-funktionale Anforderung. [25] [61] Derzeit ist das Themengebiet der Erklärbarkeit in der Wissenschaft populär, trotz dessen, dass keine allgemein anerkannte Definition der Erklärbarkeit existiert. Das steigende Interesse ist motiviert durch die zunehmende Komplexität der künstlichen Intelligenz (insbesondere durch Deep Learning Verfahren) und der ebenso zunehmenden Integrierung von Softwaresystemen in allen

Lebensbereichen. Durch die Komplexität werden Softwaresysteme undurchsichtig, sodass Endnutzer und Experten nicht nachvollziehen können auf welchen Faktoren die Ergebnisse und Verhaltensweisen beruhen. Diese Art von Systemen werden auch Black-Box Systeme genannt und gelten als *nicht erklärbar*. Durch diese Undurchsichtigkeit entsteht entweder ein Misstrauen in das Softwaresystem oder ein blindes Vertrauen in der die Ergebnisse nicht weiter hinterfragt werden. Dennoch vertrauen viele Endnutzer und Fachexperten auf die Ergebnisse dieser Softwaresysteme und hinterfragen sie nicht, was potenzielle Gefahren birgt. [34] Da die Komplexität der Softwaresysteme zunimmt und sie dadurch auch zunehmend undurchsichtig werden, besteht ein Konflikt zwischen dem natürlichen Bedürfnis nach Erklärungen der Endnutzer und der Technologie. [61] Die Erklärbarkeit stellt diese Erklärungen zur Verfügung, um Softwaresysteme durchsichtiger zu gestalten. Sie informieren den Endnutzer über die Funktionsweise des Softwaresystems, wodurch auch eine ethische, moralische und technologische Evaluierbarkeit durch das resultierende tiefergehende Verständnis ermöglicht wird. Durch das Verständnis der Systemfunktion können Endnutzer das System einschätzen und bewerten, was dazu hilft Vertrauen in das System zu schaffen. [34] Durch die gewonnene Transparenz werden ebenso menschliche Anwendungsfehler minimiert. [61] Eine Aufzählung der *Ziele der Erklärbarkeit* befindet sich in Tabelle 2.1, wobei nicht alle Ziele erfüllt werden müssen, damit ein System als erklärbar gilt. [73] Da keine allgemein anerkannte Definition existiert, wird der Begriff der Erklärbarkeit in der Literatur in einer inkonsistenten und uneinheitlichen Art verwendet. Begriffe wie Interpretierbarkeit, Überprüfbarkeit oder erklärbare künstliche Intelligenz (XAI) werden teilweise synonymartig verwendet. [61] Die Motivation ist jedoch stets dieselbe: Die Funktionsweise und die Entscheidung einer Software dem Endnutzer verständlich zu machen. [36] Chazette et al. [24] haben kürzlich eine Definition der Erklärbarkeit aufgestellt, die an verschiedene Fachgebiete und Projekte angepasst werden kann. Sie definieren erklärbare Systeme wie folgt (aus dem englischen frei übersetzt):

> *"Ein System S ist erklärbar in Bezug auf einen Aspekt X von S relativ zu einem Adressaten A im Kontext C nur dann, wenn es eine Entität E (den Erklärer) gibt, die durch die Bereitstellung eines Informationskorpus I (die Erklärung von X), A in die Lage versetzt, X von S in C zu verstehen."*

Ein Aspekt kann das System im Ganzen, die Logik oder das interne Modell mit all seinen Parametern, die Absicht, das Verhalten, die Entscheidungen, die Leistungen oder das Wissen über den Benutzer oder die Welt darstellen (siehe hierfür auch die Abbildung 2.3). [24] Der Adressat stellt hierbei den Empfänger der Erklärung, also den Endnutzer, dar. Der Kontext, indem der Endnutzer das System verwendet, beschreibt die Interaktion in Abhängigkeit einer Aufgabe oder der Umgebung. Der Erklärer ist die

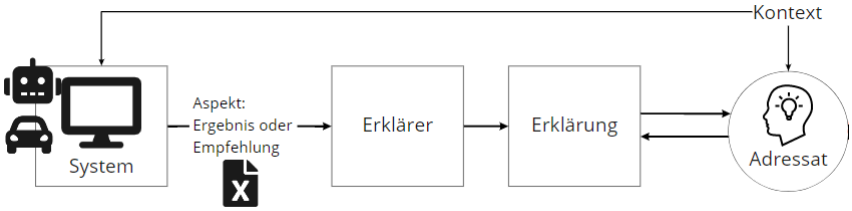

Abbildung 2.3: Funktionsweise eines erklärbaren Systems

erklärende Instanz des Systems. Er kann das System im Ganzen oder einen Teil dessen mit einem Informationskorpus erklären. Der Informationskorpus bzw. die Erklärung beinhaltet das benötigte Wissen, sodass der Endnutzer das System versteht. Durch die Erklärbarkeit eines Softwaresystems soll es dem Endnutzer daher ermöglicht werden, mithilfe von Erklärungen im vorhandenen Kontext ein Verständnis der Funktionsweise des Systems aufzubauen. Die Erklärungen stellen einen Mensch-Maschine Dialog dar und korrigieren oder schließen Lücken im *mentalen Modell* des Endnutzers. [77] Da Veständnis evaluiert werden kann, kann daran geknüpft die Erklärbarkeit eines Softwaresystems gemessen werden. [24]

Globale Erklärungen

Globale Erklärungen sind Erklärungen darüber, wie das Systemmodell insgesamt funktioniert. Dadurch werden mögliche Systemausgaben oder -verhaltensweisen dem Endnutzer näher gebracht. [73]

Lokale Erklärungen

Lokale Erklärungen sind Erklärungen, die begründen, weshalb ein bestimmtes Ergebnis ausgegeben wurde oder in welcher Art und Weise ein bestimmter Eingabeparameter die Ausgabe beeinflusst. Sie werden während der Verwendung des Systems benötigt und geben Informationen über die Berechnungen oder logischen Prozesse in einem bestimmten Kontext. [73]

2.2.1 Ziele der Erklärbarkeit

Chromik und Schuessler [29] haben anhand eines Literaturreviews eine Taxonomie der Ziele zusammengestellt, die durch die Erklärbarkeit erfüllt werden können (siehe Tabelle 2.1). Wenn mehrere Ziele erreicht werden sollen, können jedoch komplementäre, widersprechende oder unbekannte Abhängigkeiten auftreten. [29] Diese müssen innerhalb des Requirements Engineering berücksichtigt und addressiert werden.

Ziel	Definition
Transparenz	Der Endnutzer kann die Funktionsweise des Systems erklären
Überprüfbarkeit	Der Endnutzer kann die Ausgabe des Systems auf Korrektheit bewerten
Vertrauen	Das Vertrauen des Endnutzers in das System stärken
Überzeugungskaft	Der Endnutzer verwendet die Systemausgabe für seine Entscheidung
Effektivität	Dem Endnutzer helfen, gute Entscheidungen zu treffen
Effizienz	Der Endnutzer kann schneller Entscheidungen treffen
Zufriedenheit	Die Endnutzerfreundlichkeit und -erfahrung verbessern
Bildung	Dem Endnutzer die Möglichkeit geben zu lernen und zu verallgemeinern
Fehlersuche	Den Endnutzer in die Lage versetzen, Fehler im System zu erkennen

Tabelle 2.1: Ziele der Erklärbarkeit nach [29]

2.2.2 Trade-off der Erklärbarkeit

Die Erklärbarkeit kann innerhalb ihrer eigenen Ziele (die auch selbstständige Anforderungen sein können) und anderen Anforderungen in einen Konflikt geraten. [61] [25] Beispielsweise können neurale Netze genaue Ergebnisse berechnen, sind jedoch selbst für den Entwickler undurchsichtig, wodurch die Qualität der Erklärbarkeit leidet. Dies kann zufolge haben, dass einfache Modelle mit intrinsischer Erklärbarkeit verwendet werden müssen, wodurch widerrum die Genauigkeit der Berechnungen abnehmen kann. Daher muss abgewägt werden, ob die Anforderung an die Genauigkeit der Ergebnisse oder die vollständige Erklärbarkeit des Systems priorität hat. [73] Weiterhin können Wechselwirkungen auftreten, die nicht immer negativ sein müssen. Beispielsweise kann sich Erklärbarkeit positiv auf die Endnutzererfahrung auswirken, indem die Funktionsweise komplexer oder undurchsichtiger Systeme erklärt wird. Andererseits ist es auch möglich, dass sich Erklärungen negativ auf die Endnutzererfahrung auswirken, da sie unter anderem nicht benötigte Informationen darstellen, dessen Konsum dem Endnutzer Zeit kostet. [25] Da dieser Trade-off insbesondere durch nicht endnutzerzentrierte Erklärbarkeit entsteht und ebenso auch andere Endnutzeranforderungen dadurch negativ beeinflusst werden können, sollte die Erklärbarkeit endnutzerzentriert gestaltet werden. [73]

2.3 Mensch-Maschine Kommunikation

Da Erklärungen eine Kommunikationsform des Systems mit dem Endnutzer sind [77], muss das Fachgebiet der Mensch-Maschine Kommunikation (MMK) betrachtet werden. Diese forscht an der Interaktion des Endnutzers (Mensch) mit einem System (Maschine) und bietet gestalterische Lösungsmöglichkeiten. Sie erfährt viel Einflüss aus unterschiedlichen Fachbereichen wie unter anderem der Informatik, der Psychologie, der Soziologie und der Wirtschaft. Das Ziel ist es nützliche und endnutzerfreundliche Softwaresysteme zu entwickeln die gerne von Endnutzern verwendet werden. Dafür wird ein *endnutzerzentrierter Designansatz* angewendet. [1]

2.3.1 Endnutzerzentriertes Design

Endnutzerzentriertes Design ist ein iterativer Softwaredesignprozess, in dem die Bedürfnisse des Endnutzers im Fokus stehen. Dafür stehen eine Vielzahl an untersuchenden oder Ideen erzeugender Methoden zur Verfügung. Wenn Benutzererfahrungen und der Nutzungskontext modelliert werden soll, sind unter anderem relevante Methoden *Personas*, *Szenarien*, eine *Kontextanalyse* oder eine *Benutzerobservierung*. Anhand der erhobenen Informationen werden Designansätze entwickelt, die wiederrum auf die Erfüllung der Anforderungen hin evaluiert werden. Wenn beispielsweise ein konzeptionelles Design erstellt werden soll, kann dies mit einem *Prototyp* realisiert werden. Damit können ebenso Endnutzerevaluationen durchgeführt werden, um sicherzugehen, dass die Endnutzeranforderungen erfüllt sind. [93] Da es sich um einen iterativen Prozess handelt, kann anschließend beliebig in eine der vorherigen Phasen wieder übergegangen werden. Wenn aus der Evaluierung die Erkenntnis erlangt wird, dass alle Anforderungen erfüllt und die Endnutzerrückmeldungen positiv sind, kann der Prozess beendet werden. [86] Abbildung 2.4 stellt den endnutzerzentrierten Entwicklungsprozess nach der ISO-Norm 9241-210 grafisch dar. [1]

2.4 Requirements Engineering und endnutzerzentriertes Design

Endnutzer stellen einen der Stakeholdergruppen dar. Dennoch bleiben sie im Softwareentwicklungsprozess der Praxis oftmals außen vor. Da die Methoden des endnutzerzentrierten Designprozesses sich auf die Endnutzerbedürfnisse und den jeweiligen Anwendungskontext fokussieren, erweist sich eine Einbettung dieser Methoden insbesondere in das Requirements Engineering, welches dieselben Ziele mit den Stakeholdern verfolgt, als vorteilhaft. So können die Anforderungen der Endnutzer in der systematischen Art und Weise des Requirements Engineering einbezogen werden. Gleichzeitig wird

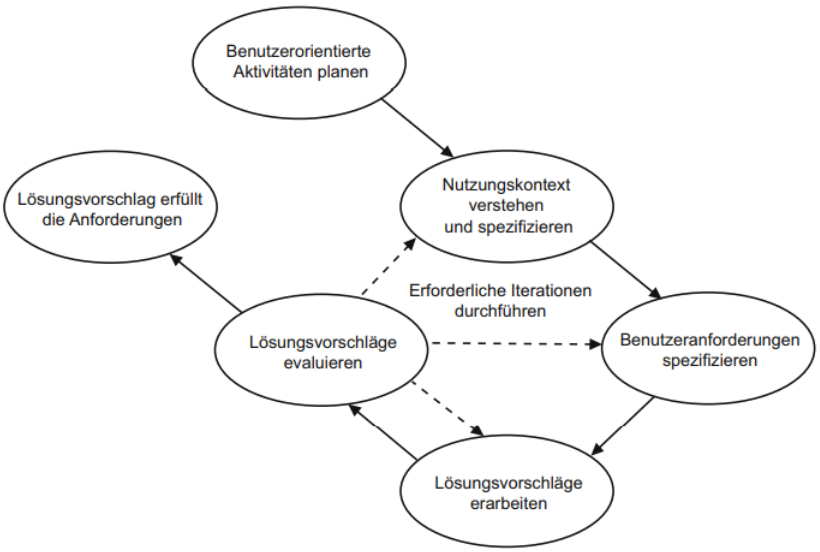

Abbildung 2.4: Das endnutzerzentrierte Prozessmodell definiert in der ISO-Norm 9241-210 entnommen aus [86]

auch die Erfüllung der Endnutzeranforderungen innerhalb eines Softwareentwicklungsprozesses sichergestellt. [86] In dieser Masterarbeit werden aufgrund dessen das Requirements Engineering mit den endnutzerzentrierten Methoden gemeinsam betrachtet, um die Forschungsfragen zu beantworten.

Kapitel 3

Verwandte Arbeiten

Wie endnutzerzentrierte erklärbare Systeme entwickelt werden können, ist in der Forschung derzeit ein aktuelles Thema. Eiband et al. [38] haben ein endnutzertentriertes Framework für die Entwicklung transparenter Benutzeroberflächen veröffentlicht, welches von Tsai und Brusilovsky [97] für die Entwicklung erklärbarer Oberflächen angewendet wurde. Jin et al. [57] haben ein endnutzertentriertes Framework für die Entwicklung von erklärbaren Systemen veröffentlicht und stellen *Explanatory Forms* vor, die den algorithmischen Aspekt des Systems beachten und zugehörige Designfaktoren darstellen. Mohseni et al. [73] haben ein Framework für den Entwurf und die Evaluation von erklärbaren Systemen entwickelt. Sie gehen darauf ein *Wie, Was* und für *Wen* erklärt werden soll und setzen Designziele, sowie Evaluationsmethoden für diese Ziele auf. Sie betrachten dabei die algorithmische Ebene, wie auch die Benutzeroberfläche. Wolf [104] empfiehlt eine szenariobasierte Entwicklung erklärbarer Systeme, auf Basis dessen Anforderungen erhoben und analysiert werden können. Cirqueira et al. [30] wenden diese für die Anforderungserhebung von Fraud-Detection Systemen an. Tjeerd et al. [91] haben ein endnutzerzentrierten Designansatz und wiederverwendbare Erklärungsmuster für erklärbare Systeme aufgestellt. Sie gehen dabei auf die Domainanalyse, Anforderungserhebung und den Entwurf, sowie die Evaluierung von multimodalen Interaktionen ein. Hall et al. [46] stellen eine systematische Methode auf, um die Anforderungen der erklärbaren Systeme zu verstehen. Weiterhin haben Gunning und Aha [45] im Rahmen des DARPA Programms ein endnutzertentriertes Evaluationsframework für erklärbare Systeme aufgestellt. Im Folgenden werden diese wissenschaflichen Arbeiten näher erläutert.

Verwandte Arbeiten der Prozessaufstellung

Eiband et al. [38] haben einen partizipativen Gestaltungsprozess zur Entwicklung von transparenten Systemen erstellt. Die wissenschaftliche Arbeit fokussiert sich auf die Tranzparenz und Verständlichkeit von

Benutzeroberflächen, welches einen Teilaspekt der Erklärbarkeit darstellt. Sie gehen dabei auf die Fragen, *Was* und *Wie* erklärt werden soll, ein. Zur Beantwortung der Fragen empfehlen sie die Anwendung unterschiedlicher Methoden. Für die Beantwortung der *Was* Frage erstellen sie drei mentale Modelle. Das *Expert Mental Model* fasst die Funktionsweise des Algorithmus zusammen und stellt das optimale *User Mental Model* dar. Das *User Mental Model* fasst das Gedankenkonstrukt des Endnutzers bezüglich der Funktionsweise des Systems zusammen. Die Differenzen beider mentaler Modelle ergeben das *Target Mental Model*, welches darstellt, welche Faktoren zusätzlicher Erklärungen bedürfen. Mit diesen Informationen wird ein erklärbarer Prototyp in iterativer Weise entwickelt und evaluiert. Bei der Evaluation mit Endnutzern wird das *User Mental Model* aktualisiert und auf eine Übereinstimmung mit dem *Target Mental Model* überprüft. Stimmen beide Modelle überein, wird davon ausgegangen, dass eine genügende Transparenz erreicht wurde. Um das Framework auf Tauglichkeit zu testen, wenden sie es für die Weiterentwicklung eines kommerziellen Softwaresystem an. Perpektiven von Endnutzern, Designern und Softwareanbietern werden miteinbezogen, um dadurch den Prozess zu validieren. Ihr Ziel ist es die mentalen Modelle der Endnutzer unter Beachtung der Stakeholderbedürfnisse zu verbessern. Ihre Ergebnisse sagen aus, dass das Framework sich zur Verbesserung der mentalen Modelle eignet, Stakeholderbedürfnisse mit einbezieht und abdeckt. Sie spekullieren, dass ihr Framework universal einsetzbar ist, da mentale Modelle in der Vergangenheit in anderen Kontexten erfasst und verbessert wurden. Sie zitieren hierfür die Werke von Kulesza et al. [65], [64], die sich mit Erklärungen für Endnutzer beschäftigen, fügen aber hinzu, dass weitere Evaluierungen hierzu benötigt werden. Sie gelangen zum Schluss, dass Benutzeroberflächen unter Beachtung der Transparenz entworfen werden sollten.

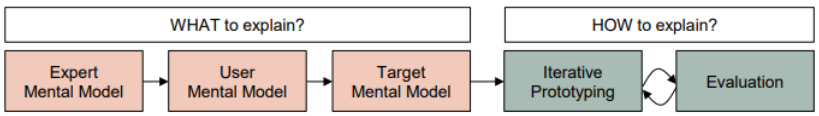

Abbildung 3.1: Darstellung des Frameworks von Eiband et al. [38] und ebenso entnommen aus dieser

Tsai und Brusilovsky [97] beschäftigen sich mit einem partizipativem Gestaltungsprozess von erklärbaren Benutzeroberflächen für ein social recommender System, welches mehrere Erklärungsziele erfüllen soll. Hierfür verwenden sie das Framework von Eiband et al. [38] innerhalb eines vierstufigen Gestaltungsprozesses. In der ersten Stufe erstellen sie ein *Expert Mental Model*, in der sie die Funktionsweisen von fünf Systemmodellen

festhalten. In der zweiten Stufe erstellen sie ein *User Mental Model*, in dem sie die Designfaktoren einer erklärbaren Oberfläche untersuchen. Dafür führten sie eine Umfrage mit 14 Teilnehmern durch, um benötigte demografische Informationen zu erfassen und eine Selbstreflexion darüber wie eine erklärbare Oberfläche in Abhängigkeit von sieben Erklärbarkeitszielen entworfen werden sollte (siehe die ersten sieben Ziele aus 2.1). In der dritten Stufe erstellen sie ein *Target Mental Model*, indem sie eine kontrollierte Laborstudie mit 15 Teilnehmern durchführten. Das *Target Mental Model* beinhaltet Informationen über die wichtigsten Komponenten, die der Endnutzer möglicherweise erklärt bekommen möchte. Sie führten eine *Card-Sorting Aufgabe* durch, um ihre Präferenz für jede der 19 gelisteten Faktoren zu erheben. In der vierten Stufe erstellen sie 25 erklärbare Oberflächen für fünf recomendation models. Sie führten eine Evaluierung mit einer within-subject-design Methodik durch, in der alle Teilnehmer eine *Card Sorting Aufgabe* lösen sollten, in der sie die vorgegebenen Oberflächen in Bezug auf einen von 19 vorgegebenen Untersuchungsfaktoren[1] in eine von fünf Gruppen zuordnen, die von *stimme voll zu* bis *stimme überhaupt nicht zu*, reicht und pro Gruppe eine absteigende Sortierung der Oberflächen vornehmen. Außerdem steht dem Teilnehmer zu eine Oberfläche als irrelevant zu markieren. Im Anschluss wurde ein semi-strukturiertes Interview durchgeführt, um qualitative Rückmeldungen zu erheben.
Ihre Beobachtungen zeigen, dass die Teilnehmer grafische erklärbare Oberflächen gegenüber den textuellen bevorzugen. Daraufhin führten sie für die erst- und zweitbeliebteste Oberfläche weitere Diskussionen mit den Teilnehmern um Rückmeldungen zu erheben. Diese zwei erklärbaren Oberflächen sollen innerhalb zukünftiger Arbeiten in ein bestehendes System implementiert werden, um sie anschließend mit Endnutzern und einer Aufgabenstellung zu evaluieren. Anhand dessen erhoffen sie sich Informationen darüber zu gewinnen, *Warum* und *Wie* Endnutzer eine erklärbare Oberfläche verwenden.

Mohseni et al. [73] haben ein Framework auf Basis eines strukturierten und iterativen Literaturreviews im Bereich des XAI aufgestellt. Sie sammelten XAI Designziele und jeweilige Evaluierungsmethoden und stellen sie dar. Dabei betrachteten sie auch die unterschiedlichen Ziele verschiedener Endnutzergruppen und stellen dabei die Fragen *Was* und *Wie* erklärt werden soll. Auf Basis der gewonnenen Erkenntnisse, stellen sie ein iteratives Design und Evaluationsframework mit mehreren Zyklen und Verschachtelungen auf, welches algorithmische und endnutzerbezogene Aspekte betrachtet. Das Framework soll dabei von multidisziplinären XAI-Teams ausgeführt werden

[1] Ein Faktor ist beispielsweise: Die Visualisierung ermöglicht es mir zu sagen, warum dieses System mir die Person empfiehlt

können. Um das Framework zu validieren, wenden sie es innerhalb einer Fallstudie an. Auf die Effektivität des Frameworks wird nicht eingegangen. Die Autoren verweisen darauf, dass weitere Fallstudien benötigt werden, um das Framework weiter zu optimieren.

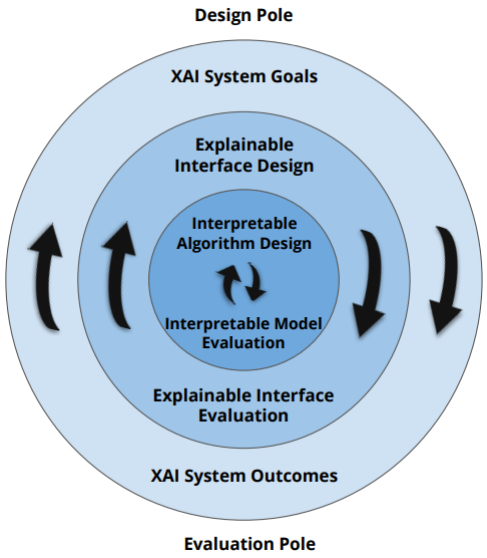

Abbildung 3.2: Darstellung des Frameworks entnommen von Mohseni et al. [73].

Verwandte Arbeiten des Requirements Engineering

Wolf [104] empfiehlt *Szenariobasiertes Design* für die Anforderungserhebung der endnutzerzentrierten XAI Systeme. Anhand von *Szenarios* kann mit einem *Brainstorming* direkter Bezug auf die realen Problematiken der Endnutzer genommen werden, was zu kreativeren und innovativeren Designentscheidungen führt. *Szenariobasiertes Design* ermöglicht insbesondere während der frühen Entwicklungsphase die Analyse des Systems in Bezug zum Kontext. Wolf hat insbesondere "Erklärbarkeitsszenarios"definiert, die speziell für die Entwicklung von erklärbaren Systemen verwendet werden sollen. Mit dieser Vorgehensweise können sich die Entwickler darauf fokusieren *Was* die Endnutzer verstehen müssen, *Welche Art* von Erklärungen die Endnutzer bei der Verwendung benötigen und *An wen* sich die Erklärungen richten. Durch die Erklärbarkeitsszenarios erhobenen Anforderungen bieten eine Grundlage, um anschließend technologische Lösungsansätze zu finden. In dem wissenschaflichen Text liegt der Fokus auf einer Methode für die Anforderungserhebung und -analyse. In dieser Masterarbeit wird eine Sammlung aller verwendeten oder empfohlenen Methoden angelegt.

Cirqueira et al. [30] verwenden, am Beispiel eines Fraud Detection XAI Systems, eine szenariobasierte Anforderungsanalyse. Dieser Ansatz umfasst folgende fünf Schritte:

- Umgebung und Kontext der Stakeholder analysieren
- Ziele der Stakeholder identifizieren
- Tools der Stakeholder identifizieren
- Alltagsrepräsentierende Szenarios auf Basis der Ergebnisse erstellen
- Anforderungen anhand der Szenarios identifizieren

Die Anforderungserhebung wurde mit drei problemzentrierten Experteninterviews eingeleitet um mehrere Meinungen über die Problematik betrachten zu können. Auf Basis der Informationen wurden Szenarios geschrieben die wieder mit den Experten auf Korrektheit validiert wurden. [30] Die erhobenen Anforderungen aus den Szenarios können anschließend in Prototypen eingearbeitet werden.

Hall et al. [46] stellen eine fünfstufige Methode auf, um die Anforderungen für den Entwurf von XAI Systemen verstehen zu können. Die Methode wurde innerhalb eines industrieellen Forschungsprojekts anhand eines semi-strukturierten Interviews mit zwölf erfahrenen Luft- und Raumfahrtingenieuren durchgeführt, die unterschiedliche technische Aufgaben, wie dem Systementwurf, der Prüfung, der Verifizierung und Validierung haben. Folgende fünf Schritte werden definiert (entnommen und frei übersetzt aus [46]):

- 1. Bestimme die relevanten Rollen innerhalb des Ökosystems.
- 2. Bestimme die relevanten Erklärungsmerkmale.
- 3. Erfasse die Erklärungsanforderungen von Personen, die den spezifischen Rollen zugehören. Dies kann durch einen traditionellen Requirements Engineering Prozess oder durch die Anwendung agiler Methoden zur Anforderungserhebung erfolgen.
- 4. Bewertung der Eignung erklärbarer Methoden zur Erfüllung der Anforderungen.
- 5. Verknüpfe bestehende erklärbare Techniken auf XAI-Systemanforderungen. Dadurch wird festgestellt, welche bestehenden Techniken bestimmte Anforderungen erfüllen, und es werden etwaige Lücken in den Fähigkeiten aufgezeigt. Dies bringt Lücken hervor,

wodurch weitere Forschung und Entwicklung für die jeweilige Anwendung gefördert wird.

Die Autoren sehen ihre Methode als einen Ansatz, welcher in weiteren Szenarien angewendet werden soll, um ihn weiter zu verbessern. Sie gehen dabei nicht explizit auf MMK Methoden ein.

Nguyen et al. [79] empfehlen einen dreistufigen Requirements Engineering Prozess am Beispiel von IoT Cloud Systemen. Sie betrachten die Anforderungen an die Erklärbarkeit dabei ganzeinheitlich und beziehen mehrere Stakeholdergruppen ein. Zuerst sollen konkrekte Stakeholder ermittelt werden, deren Anforderungen anhand von Interviews oder Befragungen erfasst werden. Die gesammelten Anforderungen werden in einem *Explainability Requirement Template* festgehalten, welches die Aspekte der Erklärbarkeit, den Prozess, die Quelle der Anforderung, den Stakeholdertyp, den Grund der Erklärung und die Priorität jeweils in der *ExplainabilityList* abdeckt. Abschließend empfehlen sie eine kontinuierliche Aktualisierung der Anforderungen im Laufe der Entwicklung.
Ihre Methode wendeten sie mit einer Fallstudie in der Telekommunikationsbranche an. Es konnte eine Fülle an Anforderungen gesammelt werden, um anschließend angemessene Lösungen zu definieren. [79]

Abgrenzung

Der Fokus der verwandten Arbeiten liegt auf dem Prozess für die Entwicklung erklärbarer Systeme. Sie stellen dar, wann welche Aktivität durchgeführt werden soll. Methoden werden entweder nur beispielhaft genannt oder nur für die Evaluationsaktivität detaillierter beschrieben. Keine der Arbeiten betrachtet die Methoden, die für die Entwicklung erklärbarer Systeme benötigt werden, Ende-zu-Ende. Im Rahmen dieser Masterarbeit werden insbesondere die Menge der Methoden betrachtet, einer Aktivität zugeordnet und die Art und Weise der Anwendung der Methode bezüglich erklärbarer Systeme erläutert. Während einige Arbeiten Aktivitäten für die Entwicklung erklärbarer Systeme beschreiben, gehen sie nicht detaillierter darauf ein, welche Methoden dafür geeignet sind und wie diese ausgeführt werden sollen. Wiederrum in andere Arbeiten werden Methoden empfohlen oder vorgeschlagen, die aber nicht in den Kontext des Requirements Engineering gesetzt werden. Zusammengefasst exisiteren viele Ansätze, die jedoch noch nicht im Gesamten als eine Einheit betrachtet werden. Diese Arbeit hat das Ziel einen Workflow aufzustellen, der auch im industrieellen Kontext anwendbar ist, durch die Anwendung der bereits etablierten Methoden aus dem Requirements Engineering und der Mensch-Maschine Kommunikation, um die Hürde für die Entwicklung erklärbarer Systeme niedrig zu halten. Die hier dargestellten Arbeiten werden ebenso für den Aufbau des erklärbaren Systems verwendet.

Kapitel 4

Methodik

Zur Beantwortung der Forschungsfragen wurde ein Mixed-Method Verfahren verwendet. Zunächst wurden anhand eines Literaturreviews Methoden und Aktivitäten gesammelt, welche in der Literatur für Erklärbarkeit verwendet oder vorgeschlagen werden. Basieren auf der Ergebnisse wurde ein Workflow für erklärbare Systeme entworfen. Da dieses Vorgehen nicht ausreicht, um den Workflow auf ihre Tauglichkeit zu bewerten, wurden anschließend Experteninterviews durchgeführt. In den Experteninterviews wurden verwendete und vorgeschlagene Methoden und Aktivitäten für die Entwicklung von erklärbaren Systemen ermittelt. Mit den Ergebnissen aus den Interviews wird der literaturbasierte Workflow evaluiert und gegebenenfalls optimiert. Der detaillierte Ablauf wird im Folgenden beschrieben.

4.1 Literaturreview

Es wurde ein systematisches Literaturreview nach dem Leitfaden von Boland et al. [14] durchgeführt. Zuerst wurden unterschiedliche Suchstrings getestet, wodurch auch weitere Fachbegriffe gefunden wurden, die in den finalen Suchstring integriert wurden. Der final definierte Suchstring wurde in der Suchmaschine für wissenschaftliche Texte, *Google Scholar*, angewendet. Anschließend wurde eine Literatursuche mit der *technischen Informationsbibliothek Hannover* durchlaufen. Hierfür wurde folgender Suchstring definiert:

(“explainability“ OR “explainable“ OR “explanation“ OR “explaining“ OR “XAI“) AND (“system“ OR “software“ OR “design“ OR “interface“ OR “HCI“ OR “human-computer interaction“ OR “RE“ OR “requirements engineering“ OR “SE“ OR “software engineering“)

Der Suchstring verbindet mit dem *AND*-Operator Stichwörter der Erklärbarkeit und Softwareentwicklung oder -design miteinander. Dadurch

konnten wissenschaftliche Texte über die Entwicklung und der Designentscheidungen für erklärbare Systeme gesammelt werden, die für die Beantwortung der Forschungsfragen notwendig sind. Das Themengebiet der Erklärbarkeit ist multidisziplinär. Neben der Informatik beschäftigen sich Fachbereiche aus der Mathematik, Psychologie, Philosophie und Ethik mit der Erklärbarkeit. [101] Die Suchergebnisse mussten daher auf Bezug zur Informatik selektiert werden. Dies geschah hauptsächlich über den Titel, bei Uneindeutigkeiten wurde auch die Zusammenfassung gelesen und anschließen selektiert. Weiterhin wurden wissenschaftliche Texte die erklärbare Systeme entwickelt oder an ihnen geforscht haben ausselektiert, wenn sie keine Angaben über die verwendeten Softwareentwicklungsmethoden und -aktivitäten getroffen haben. Die Resultate wurden in folgender Reihenfolge mit folgenden Ausschluskriterien selektiert:

- Kein Zugriff mit der Universitätslizenz (20 Resultate)
- Kein Bezug zum Gebiet der Informatik (31 Resultate)
- Nicht in englischer oder deutscher Sprache verfasst (4 Resultate)
- Keine Erwähnung von Aktivitäten oder Methoden (275 Resultate)

Innerhalb des Literaturreviews wurden insgesamt 446 wissenschaftliche Texte analysiert, von denen 116 als relevant für die Beantwortung der Forschungsfragen beurteilt wurden. Ein Text wurde als relevant eingestuft, wenn Methoden oder Aktivitäten verwendet oder vorgeschlagen wurden, um erklärbare Systeme zu entwickeln. Die Ergebnisse wurden in einer Konzeptmatrix zusammengefasst und nach Häufigkeit der verwendeten oder vorgeschlagenen Methoden und Aktivitäten absteigend sortiert. Auf Basis der Konzeptmatrix, sowie Arbeiten mit verwandter Thematik, wurde ein Workflow zur Entwicklung von erklärbaren Systemen entworfen.

Die Literaturrecherche wurde durchgeführt bis eine Sättigung der Ergebnisvielfalt erreicht wurde. Die Sättigung wurde mit 50 aufeinanderfolgenden Texten definiert, aus der keine neuen Erkenntnisse gewonnen werden konnten. Die untersuchten wissenschaftlichen Arbeiten wurden gesammelt und in einer Konzeptmatrix mitsamt der empfohlenen oder verwendeten Methoden und Aktivitäten zusammengefasst.

4.2 Experteninterviews

Im zweiten Schritt wurden 19 semistrukturierte explorative Experteninterviews durchgeführt, mit einer Dauer von jeweils einer Stunde. Mit semistrukturierten Interviews können neue Erkenntnisse und Informationen innerhalb eines vorgegebenen thematischen Rahmens erhoben werden. Der Rahmen wurde mit zwei vorbereiteten Aufgaben und vorgeplanten Fragen gelegt.

Die Interviews wurden online im *BigBlueButton* und, als Ausweichportal bei technischen Schwierigkeiten, *Jitsi Meets* durchgeführt. Das Gespräch wurde aufgezeichnet und anschließend transkribiert um es auswerten zu können. Da bei einem reinen Gespräch Informationen vergessen werden können und es schwer ist den Überblick zu halten, wurde das kollaborative Zeichentool *Miro* verwendet, um den Softwareentwicklungsprozess grafisch festzuhalten. Anhand einer Erklärungsphase wurde die Thematik der Erklärbarkeit mit den Teilnehmern besprochen, da die nicht-funktionale Anforderung den Teilnehmern noch recht unbekannt war. Anschließend wurde die Problematik von fehlender Erklärbarkeit anhand eines Szenarios und einer daran gekoppelten Aufgabe dem Teilnehmer verdeutlicht. Dadurch wurde ein *Brainstorming* angeregt. Es wurden zwei Aufgaben im Miro Board grafisch vorbereitet, die mit der *Think-Aloud* Methodik unterstützend bearbeitet wurden.

Zur Auswertung wird die Inhaltsanalyse nach Mayring angewendet. [71] Dieser kodierende Auswertungsprozess ermöglicht die systematische Erschließung von Interviewtranskripten in qualitativer Form, wobei durch das Kodierungsverfahren quantifizierende Auswertungen angewendet werden können. Da mit den Experteninterviews in erster Linie qualitative Ergebnisse gewonnen werden, zur Beantwortung der Forschungsfrage aber auch quantitative Ergebnisse benötigt werden, eignet sich die Inhaltsanalyse nach Mayring zur Beantwortung der Forschungsfrage. Die 19 einzelnen Interviewtranskripte haben insgesamt 95000 Zeilen, wobei zwei Interviews auf Englisch gehalten wurden. Die Kategorisierung der Aussagen in den Interviewtranskripten erfolgt deduktiv, das heißt, dass die Kategorien theoretisch fundiert in der Literatur bereits vorhanden sind. Im Rahmen dieser Masterarbeit werden die Methodiken aus der Mensch-Maschine Kommunikation, des Requirements Engineering oder des Software Engineering als Kategorien definiert. Gezählt wurden die Angaben nur, wenn die Kategorie explizit vom Experten erwähnt wurde. Die Aufgabenstellung wurde so entworfen, sodass eine explizite Nennung der Methoden verpflichtend ist. Eine Reliabilitätsprüfung mit einer zweiten Person war einerseits nicht möglich, andererseits aber auch nicht von großer Bedeutung, da die Kategoriesierung der Experte durch die Aufgabenstellung direkt selbst vornahm.

4.2.1 Pilotinterviews

Um das Interviewdesign zu testen wurden drei Pilotinterviews mit Mitarbeitenden aus dem Fachgebiet des Software Engineerings durchgeführt. Das erste Interviewdesign umfasste die erste Aufgabe in mündlicher Form, die zweite Aufgabe hatte lediglich Unterschiede in der visuellen Darstellung, die eine Sequenz suggerierten. Anschließend war eine Meinungseinholung der Experten für den literaturbasierten Workflow geplant. Das erste Pilotinterview fokussierte sich auf die Bewertung des Interviewdesigns und weniger auf die genaue Durchführung. Es wurde vorgeschlagen die sequenzartige

Darstellung zu entfernen, da sie das Erebnis vordefinieren würde, wehsalb diese durch voneinander unabhängige Blasen ersetzt wurden. Außerdem wurde empfohlen die direkte Bewertung des literaturbasierten Workflows durch die Experten nicht durchzuführen, weshalb dieser Schritt aus dem Interviewdesign entfernt wurde. Dafür sollte der aktuelle Workflow der Experten innerhalb ihres Unternehmens zusätzlich grafisch festgehalten werden, um ein Abgleich durchführen zu können. Weiterhin wurde die FLOW-Notation vorgeschlagen, welche sich jedoch auf die Informationsflussanalyse konzentriert. Diese Komponente zusätzlich zu erfassen hätte den zeitlichen Rahmen der Interviews gesprengt und wurde daher verworfen. Die anderen Vorschläge wurden eingearbeitet und zwei weitere Pilotinterviews wurden durchgeführt. Die Ergebnisse aus den Pilotinterviews waren großteils positiv, aber es wurden Bedenken bezüglich der Schwierigkeit der Aufgaben geäußert. Der von den Pilotteilnehmern entworfene erklärbare Workflow wurde als nicht realitätstauglich bewertet. Dadurch wurden Fragen für die Experten vorbereitet, die diesen Fall abdecken sollen, um die Gründe hierfür zu analysieren.

4.2.2 Teilnehmersuche

Das Interview ist von explorativer Art, da die Erklärbarkeit eine recht neue Anforderung ist, die derzeit innerhalb von Unternehmen noch keiner großen Bekanntheit unterliegen ist. In explorativen Interviews werden Teilnehmer nicht unter den Aspekten statistischer Repräsentativität gewählt, sondern unter Beachtung der Eigenschaften der Person. Es sollen vielmehr die Ideen, Meinungen und Erfahrungen erhoben werden, wie die Teilnehmer die Anforderung der Erklärbarkeit anhand ihres Wissens aus der Industrie umsetzen würden. Im Vorfeld wurden daher Teilnahmekriterien definiert, um Gesprächspartner zu rekrutieren, die über die erwähnten Eigenschaften verfügen. Teilnehmer sollten folgende Rollenbeschreibungen haben: *Product Owner, Requirements Engineer, Software Developer oder Software Engineer*. Es wurden gezielt nach diesen Rollenbeschreibungen gesucht, da diese Experten dem Problem der Erklärbarkeit in ihrem Arbeitsalltag potenziell unterworfen sind oder sein können. Zum Interview wurden 87 Experten über unterschiedliche Kommunikationskanäle eingeladen. Über den Social-Media-Kanal *LinkedIn* wurden 43 Anfragen versendet. Über persönliche Kontakte wurden weitere 35 Personen telefonisch, per E-Mail und persönlich angefragt. Weiterhin wurde die Kontaktinformation zu einem Unternehmen von der Betreuerin dieser Masterarbeit bereitgestellt, welches sich im vorhinein bereiterklärte teilzunehmen. Das Unternehmen stellte neun Teilnehmer bereit. Die Rücklaufquote lag somit bei 22 %, wobei die meisten Anfragen nicht beantwortet wurden. Ablehnungen wurden begründet mit Zeit-/Ressourcenmangel, Abwesenheit aufgrund von Urlaub oder einschränkenden Unternehmensrichtlinien. Schließlich wurden 19 Teilnehmer aus sieben

Unternehmen einzeln in einem einstündigen semistrukturierten Experteninterview befragt. Da den Teilnehmern Anonymität versichert wurde, werden die erhobenen Teilnehmerdaten nur im Groben dargestellt. Tabelle 4.1 listet alle Teilnehmerdaten auf. Da in der Realität die Rollenbeschreibungen der Teilnehmer oft anders und viel spezifischer sind, im Vergleich zu den vordefinierten Rollen, wurde die Einteilung vom Autor thematisch durchgeführt, auch um die Anonymität der Teilnehmer zu wahren. Kleine Unternehmen wurden mit weniger als 50 Mitarbeitenden, mittlere mit weniger als 250 und große mit mehr als 250 Mitarbeitenden definiert.

ID	Alter	Rolle	Berufsjahre	Unternehmensgröße
1	37	Requirements Engineer	17	Klein
2	32	Product Owner	5	Klein
3	27	Requirements Engineer	2	Klein
4	23	Softwaredeveloper	3	Klein
5	32	Softwaredeveloper	7	Klein
6	31	Requirements Engineer	5	Klein
7	27	Softwaredeveloper	2	Klein
8	33	Softwaredeveloper	9	Klein
9	25	Softwaredeveloper	4	Klein
10	35	Requirements Engineer	8	Groß
11	34	Softwaredeveloper	10	Mittel
12	28	Softwaredeveloper	6	Mittel
13	35	Softwaredeveloper	3	Groß
14	29	Product Owner	5	Groß
15	41	Product Owner	16	Groß
16	34	Product Owner	2	Mittel
17	35	Requirements Engineer	8	Groß
18	35	Requirements Engineer	2	Mittel
19	39	Product Owner	12	Mittel

Tabelle 4.1: Teilnehmer der Interviewstudie

4.2.3 Leitfaden

Im Folgenden wird der Ablauf des Interviews dargestellt[1]:

1. Gesprächseröffnung: Vorstellung
2. Informationen über Thematik und Ablauf
3. Visuelles Zeichentool öffnen:

[1] Eine grafische Darstellung der einzelnen Aufgaben, der Erläuterung der Erklärbarkeit und des Fallbeispiels findet sich im Anhang

4. Erklärung und Bearbeitung der Aufgabe 1: Darstellung des aktuellen Softwareentwicklungsprozesses im Unternehmen

5. Erläuterung der Erklärbarkeit: Darstellung einer formalen Definition und eines Negativbeispiels mit textueller und grafischer Darstellung

6. Fallbeispiel: Ein Szenario über die geplante Entwicklung eines Systems an das die Anforderung der Erklärbarkeit gestellt wird. Der Teilnehmer versetzt sich in die Rolle eines Prozessbeauftragten und soll einen Prozess aufstellen, der die Entwicklung des erklärbaren Softwaresystems ermöglicht

7. Erklärung und Bearbeitung der Aufgabe 2: Darstellung eines Softwareentwicklungsprozesses für erklärbare Softwaresysteme

   (a) Frage 1: Kommunikationsform und -bedarf der Front- und Back-End Entwickler klären

   (b) Frage 2: Anwendbarkeit in der Realität abklären

4.2.4 1. Aufgabe: Workflows der Teilnehmer

Zunächst wurden die Experten gebeten den Softwareentwicklungsprozess in ihrem aktuellen Unternehmen darzustellen. Hierfür wurde ein Miro Board verwendet, indem der Autor und der Teilnehmer kollaborativ arbeiteten. Das Ergebnis wurde vom Teilnehmer im Board visuell dargestellt. Der Teilnehmer wurde gebeten nach der Think-Aloud Methode seine Gedankengänge und Motivation zu erklären. Um den Teilnehmern eine Gedächtnisstütze zu bieten, wurden beispielhafte Methoden und Aktivitäten aus dem Fachgebiet der MMK und dem RE aufgelistet (siehe Abbildung **??**). Die Aktivitäten wurden dem Teilnehmer mündlich erklärt, während die Methoden bei Bedarf textuell oder mündlich erklärt wurden. Um eine daraus folgende Voreingenommenheit zu reduzieren, wurde mehrmals ausdrücklich erwähnt, dass es sich nur um Beispiele handelt und der Teilnehmer keinen der aufgezählten Methoden und Aktivitäten verwenden muss. Der Teilnehmer wurde ermutigt selbst neue Methoden und Aktivitäten zu erstellen, die seinen Prozessablauf genauer beschreiben. Trotz dessen kann nicht ausgeschlossen werden, dass Informationen vergessen wurden, da nicht jeder Experte im Detail weiß, welche Methoden in anderen Aktivitäten verwendet werden und es eine zeitliche Restriktion von 20 bis 25 Minuten gab.

4.2.5 Erläuterung der Erklärbarkeit

Da den Teilnehmern die Erklärbarkeit nicht bekannt war, wurde ihnen in einem zweistufigen Verfahren die Thematik näher erläutert. Zunächst wurde die Definition von erklärbaren Systemen von Chazette et al. [24] vorgestellt

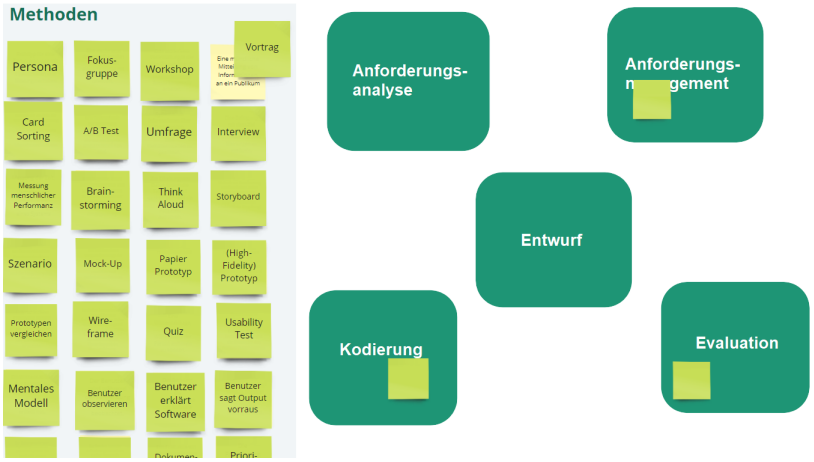

Abbildung 4.1: Beispielhafte Darstellung der Methoden und Aktivitäten. Hinter jedem Post-It verbirgt sich eine Erklärung zur Methode

und bei Bedarf diskutiert. Die Definition wurde gleichzeitig auch in einer grafischen Version dargestellt, da innerhalb eines Interviewkontexts sich das Verstehen einer formalen Definition als problematisch aufweisen kann. Falls die formale Darstellung für den Teilnehmer nicht nachvollziehbar ist, wurden die Gedankengänge des Adressaten zusätzlich in einem roten Kasten dargestellt, sodass die Vorteile der Erklärbarkeit ersichtlich sind. Um das Verständnis zu erleichtern, wurde zusätzlich ein bekanntes Negativbeispiel vorgestellt.
Im zweiten Schritt wurde die Problematik anhand eines Fallbeispiels verdeutlicht. Die Teilnehmer wurden gebeten sich in die Rolle eines Prozessentwicklers in einem fiktiven Unternehmen hineinzuversetzen, in welchem ein erklärbares autonomes Fahrzeug entwickelt werden soll. Es wurde ein autonomes Fahrzeug als Beispiel gewählt, da die Experten im Kontext der Fahrzeugbranche beschäftigt sind. Außerdem sind autonome Fahrzeuge derzeit ein populäres Thema, wodurch mögliche Probleme und Vorteile den Teilnehmern gegebenenfalls bewusst sind. Diese vorbereitende Phase diente dazu, ein allgemeines Verständnis über die Problematik zu erzeugen und die Teilnehmer auf die zweite Aufgabe vorzubereiten.

4.2.6 2. Aufgabe: Workflow für erklärbare Systeme

In der zweiten Aufgabe sollten die Teilnehmer einen Softwareentwicklungsprozess speziell für erklärbare Systeme entwerfen. Die zweite Aufgabe wurde von der Struktur genauso wie die erste Aufgabe gehalten, damit die Teilnehmer die Aufgabenstruktur nicht erstmal verstehen müssen und keine Zeit verschwenden wird. Dadurch wurden die Ergebnisse aus Aufgabe eins und zwei auch vergleichbar.

Kapitel 5

Workflow für erklärbare Systeme

Dieses Kapitel stellt die Ergebnisse aus dem Literaturreview dar, welches den aktuellen Stand der Forschung betrachtet. Das Literaturreview wurde durchgeführt, um verwendete oder empfohlene Methoden für die Entwicklung von erklärbaren Systemen zu erheben. Die Motivation war die Methoden in geeignete Phasen des Requirements Engineering einzuordnen. Da aus dem Literaturreview deutlich wurde, dass eine Evaluation für die Entwicklung erklärbarer Systeme unabdingbar ist, wurden Aktivitäten definiert, die über das Requirements Engineering hinaus gehen. Hierfür wurden die einzelnen Phasen des endnutzerzentrierten Prozessmodells (siehe Abbildung 2.4) herangezogen. Die Ergebnisse aus dem Literaturreview werden zusammengefasst und geordnet, sodass ein Workflow entsteht, welches auf dem aktuellen Stand der Forschung basiert. Strukturiert wird dieses Kapitel wie folgt: Pro Aktivität wird eine Sektion erstellt, die benötigte Konzepte zur Entwicklung erklärbarer Systeme definiert. Anschließend werden geeignete Methoden aufgelistet und erläutert, welche für die Entwicklung erklärbarer Systeme angewendet wurden. Dies soll jedoch nicht aussagen, dass nicht aufgelistete Methoden sich nicht für die Entwicklung erklärbarer Systeme eignen würden. Hierzu wurden lediglich innerhalb des Literaturreviews keine Quellen identifiziert.

5.1 Lösungsansätze

Endnutzerzentrierter Ansatz

Ein endnutzerzentrierter Entwicklungsprozess ist für die Entwicklung von erklärbaren Systemen essenziell und wird von vielen Autoren empfohlen (siehe Tabelle 5.4). Erklärbare Systeme sollen unter anderem vertrauens-

würdig, sowie verständlich für die Endnutzer sein. Um den Endnutzern ein zuverlässiges erklärbares Softwaresystem bieten zu können, muss dieses an die Bedürfnisse der Endnutzer angepasst werden. Weigand et al. [103] schlagen hierfür konkret den endnutzerzentrierten Entwicklungsprozess definiert nach der ISO 9241-210 vor. Diese Norm bildet das Grundkonzept für den erklärbaren Workflow und alle weiteren Überlegungen innerhalb dieser Arbeit. Ein erklärbares System verwendet Erklärungen die Wissenslücken füllen. Da diese sich stark spezifisch für jedes Individuum unterscheiden, spielen menschliche Faktoren eine große Rolle in der Entwicklung von erklärbaren Systemen. Die Erklärungen bilden eine Kommunikationsschnittstelle zwischen Mensch und Maschine, weshalb auch den Methodiken und Erfahrungen aus der MMK innerhalb des Workflows Beachtung geschenkt wird. [77] Um die Fülle an Anforderungen für erklärbare Systeme eingrenzen zu können, werden Endnutzergruppen spezifiert. Wird kein endnutzerzentrierter Entwicklungsprozess angewendet, kann dies zu Folge haben, dass die Erklärungen unwirksam oder sogar potenziell risikoreich sind. Letztendlich besteht die Gefahr, dass Endnutzer dem System oder den Erklärungen gegenüber keine Beachtung schenken werden. [4]

Endnutzerzentrierter Ansatz: Co-Design

Weiterhin findet das *Co-Design* mit Endnutzern viel zuspruch (siehe Tabelle 5.4). Hierbei nehmen die Endnutzer direkt am Entwicklungsprozess teil, wodurch sie eigene Ideen einbringen und Designentscheidungen während der Entwicklung evaluieren können. Dadurch können Endnutzerbedürfnisse erfasst und erfüllt werden, was die Akzeptanz der Endnutzer gegenüber dem System erhöht. Durch Diskussionen darüber wie Erklärungen in das System integriert werden können, werden Designentscheidungen konzipiert und skizziert. Die Endnutzer können das System testen und beispielsweise Mock-Ups so anpassen, wie sie es für angemessen halten. Dabei wird auch ein Brainstorming mithilfe der visualisierten Ergebnisse angeregt. Da diese Informationen allein durch Gespräche nur schwer erfasst werden können, hilft die Co-Design Methode bei der Entwicklung innovativer Entwürfe. [78] Co-Design kann Anforderungen aufdecken, Arbeitsabläufe, Ziele, Gedankengänge und Bedürfnisse der Endnutzer verdeutlichen und Trade-offs der Anforderungen untereinander aufzeigen.

Endnutzerzentrierter Ansatz: Feedback-Loop

Durch eine Zusammenarbeit von Entwicklern und Endnutzern können die Bedürfnisse der Endnutzer innerhalb eines Softwareentwicklungsprozesses genauer adressiert werden. Ein Feedback Loop ist nötig um iterativ das erklärbare Softwaresystem mit Evaluationen aus Endnutzertests verfeinern zu können (siehe Tabelle 5.4). Die Qualität des erklärbaren Systems hängt

schlussendlich von einer angemessenen Anforderungsanalyse und geeigneten Evaluationsmethoden ab, dessen Ergebnisse in die Software wieder eingearbeitet werden müssen. Die Ergebnisse aus der Evaluation geben pro Iteration neue Anhaltspunkte über die derzeitige Güte der Erklärbarkeit, welche Anforderungen erfüllt wurden und an welchen Punkten noch Verbesserungsbedarf besteht. Die ISO 9241-210 definiert ebenso einen iterativen Softwareentwicklungsprozess mit einer Evaluierungsaktivität, weshalb ein Feedback-Loop in den Workflow integriert wird.

Solution-First

Bei dem Solution-First Ansatz wird zuerste eine technische Lösung vorgeschlagen und implementiert. Erst anschließend wird das System analysiert, um die Nutzeranforderungen abzuleiten. [104] Beim Solution-First Ansatz besteht nicht die Möglichkeit während der Entscheidungsfindung Nutzerfeedback einzuholen. Dies erschwert Anpassungen des Systems, da die reine Planung des Softwaresystems sehr zeitaufwändig ist und dadurch die Hürde für Änderungen groß ist. Oftmals halten die beteiligten Personen sich an schon getätigten Entscheidungen fest. [57] [104] Es empfiehlt sich daher ein kleinschrittiger Entwicklungsprozess, indem Designansätze evaluiert werden bevor sie implementiert werden, um schnell und angemessen auf die Bedürfnisse der Endnutzer reagieren zu können. Aus diesen Gründen wird von einem Solution-First Ansatz für die Entwicklung erklärbarer Systeme abgeraten.

Lösungsansatz	Quelle
Endnutzerzentriert	[78] [99] [26] [25] [102] [40] [56] [31] [108] [105]
Co-Design	[78] [73] [109] [20] [102] [57] [15] [96] [105]
Feedback Loop/Iterativ	[73] [106] [20] [26] [44] [57] [38] [67] [42] [45] [108]

Tabelle 5.1: Literaturergebniss für die Grundstruktur des Workflows

5.2 Requirements Engineering für die Erklärbarkeit

5.2.1 Anforderungserhebung und -analyse

In der Anforderungserhebung wird der Kontext spezifiziert und die Stakeholder analysiert. [39] In dieser Aktivität muss geklärt werden, ob eine Anforderung für die Erklärbarkeit besteht. Dafür werden Stakeholdergruppen und der Anwendungskontext identifiziert und analysiert. Folgt aus der Analysephase, dass die Erklärbarkeit für die Erfüllung der Bedürfnisse und Ziele der Stakeholder benötigt wird, entstehen neue Fragen. Für *Wen* und *Was* soll erklärt werden? Zur Beantwortung dieser Fragen wird erfasst, für

welche Stakeholder die Erklärungen bestimmt sind und welche Informationen sie überbringen sollen. Ein entscheidender Faktor für die erfolgreiche Durchführung der im Folgenden aufgelisteten Aktivitäten und Methoden sind gute Kommunikationsfähigkeiten der ausführenden Rolle. Oftmals ist dieser wichtige Faktor den Entwicklern nicht bewusst, wodurch die Qualität der erhobenen Anforderungenen leidet. Beispielsweise werden Formulierungen der Fragen für Interviews oder Fragebögen leichtfertig erstellt, obwohl sie ein wichtiges Kommunikationsmittel innerhalb der Methode darstellen. [74] Daher kann ein ausschlaggebender Faktor für die Anforderungserhebung und -analyse die Anteilnahme von Softwaredesignern sein, welche Erfahrungen bei der Durchführung von qualitativen Methodiken für die Erhebung von Nutzerbedürfnissen haben. [48] Weiterhin sind Schulungen der Entwickler eine Möglichkeit das Feingefühl für die Kommunikation zu schärfen. [74] In Abbildung 5.1 wird der Workflow und die Struktur dieses Unterkapitels dargestellt.

Abbildung 5.1: Workflow für die Anforderungserhebung und -analyse

5.2.2 Produktvision

Im Rahmen der Anforderungserhebung werden die Anforderungen der Stakeholder erhoben, um sie verstehen und analysieren zu können, aus der sich eine Produktvision ableiten lässt. Die Identifizierung der Anforderung an die Erklärbarkeit sollte dabei so früh wie möglich erfolgen, um dadurch motivierte Methodiken zur Anforderungserhebung durchführen zu können. Es kann sich herrausstellen, dass die Erklärbarkeit *nicht hilfreich* für die Stakeholder ist. Dies kann beispielsweise der Fall bei spezieller Backend-Software sein, die nur von einer kleinen Expertengruppe bedient wird. Es darf bei der Anforderungserhebung aber nicht erwartet werden, dass beispielsweise der Auftraggeber die Erklärbarkeit direkt fordert. Die Erklärbarkeit ist eine noch recht unbekannte Anforderung, beispielsweise im

Vergleich zur Benutzerfreundlichkeit. Daher müssen die Ziele und Wünsche des Auftraggebers und anderer Stakeholder genau analysiert werden, ob nicht eine indirekte Anforderung an die Erklärbarkeit besteht, demm oftmals erwähnen Auftraggeber nicht-funktionale Anforderungen nicht, erwarten sie schlussendlich jedoch. [2] Dafür muss analysiert werden, inwiefern die Erklärbarkeit einen Mehrwert für die Stakeholder bietet. Rosenfeld und Richardson [88] definieren drei Stufen für die Notwendigkeit der Erklärbarkeit: *Nicht hilfreich, Nützlich* oder *Kritisch*. Bunt et al. [18] untersuchten beispielsweise, ob Erklärungen für Endnutzer in jedem intelligenten System immer notwendig sind. Ihre Ergebnisse zeigen, dass in einigen Fällen die Kosten für Erklärungen zum Beispiel für Amazon- und YouTube-Empfehlungen deren Vorteile überwiegen könnten, was die Erklärbarkeit als *nicht hilfreich* einstufen würde. Agentensysteme, wie beispielsweise Roboter die vollständig vom Endnutzer kontrolliert werden, profitieren eher nicht von der Erklärbarkeit da die Endnutzer die vollkommene Kontrolle über das System verfügen. [88] Bei Robotern die auserhalb des Sichtbereichs des Endnutzers verwendet werden, können Endnutzer von Erklärungen jedoch profitieren. Der Roboter erklärt beispielsweise seinen Ort, technische Probleme oder Probleme auf dem Gelände. Hierbei kann die Erklärbarkeit *nützlich* sein. [88]
Aber nicht nur Endnutzer und Auftraggeber können eine Anforderungsquelle an die Erklärbarkeit definieren. Weiterhin können Faktoren des Umfelds erklärbare Systeme fordern. Beispielsweise kann aus rechtlichen Gründen, wie das "Recht auf Erklärung", die Erklärbarkeit vom Gesetzesgeber zukünftigt gefordert werden oder es existiert ein ethisch motivierter Anspruch an die Transparenz eines Softwaresystems, besonders in kritischen Kontexten wie der Strafverfolgung oder des Gesundheitswesens, was sie zu potenziellen Stakeholdergruppen definiert. [25] Wenn die Erklärbarkeit dadurch motiviert benötigt wird, kann die Notwendigkeit als *kritisch* bezeichnet werden, da das Softwaresystem rechtlich gesehen entweder nicht angewendet werden dürfte oder die Gefahr einer Anwendung zu groß ist.
Weiterhin kann die Erklärbarkeit andere Aspekte des Softwaresystems *verbessern*, wie unter anderem die Transparenz, die Vertrauenswürdigkeit, die Ethik oder die Gesetzeskonformität. Weiterhin kann die Erklärbarkeit Aspekte wie die Fairness, die Endnutzerzufriedenheit, die Überzeugungskaft, die Annahme oder Akzeptanz des Softwaresystems *fördern*. Erklärbarkeit kann aber auch Aspekte *verhindern* oder *vermindern*. Wie Verwunderungen des Endanwenders, Risikos oder Biases welche motiviert durch die Systemausgaben oder -verhaltensweisen sein können. [41] Ferreira und Monteiro [41] haben eine Literaturrecherche durchgeführt, in der sie weitere Aspekte auflisten. Zusammenfassend stimmen ihre Ergebnisse mit den *Zielen der Erklärbarkeit*, aus Tabelle 2.1, überein. Ist die Motivation für die Erfüllung eines dieser Ziele vorhanden, kann dies mit der Erklärbarkeit erreicht werden. Daher können die *Ziele der Erklärbarkeit* als Kriterien zur Beantwortung

der Frage, ob das Softwaresystem Erklärbarkeit benötigt, verwendet werden. Um die Frage nach der Notwendigkeit zu beantworten, müssen Stakeholder analysiert und Endnutzergruppen bestimmt werden. Dafür können Methoden wie beispielsweise *Interviews* und *Endnutzerbeobachtungen* im Kontext der Softwareanwendung, mit den Stakeholdern durchgeführt werden. Die geeigneten Methoden werden am Ende dieses Unterkapitels aufgelistet und erläutert.

5.2.3 Zieldefinition

Zusammengefasst sollten in der Anforderungserhebung und -analyse folgende Fragen gestellt und beantwortet werden [91]:

- Wer wird das System nutzen?
- Welche typischen Aufgaben führen diese Endnutzer aus?
- Welcher Nutzen wird von der Nutzung des Systems erwartet?
- Was ist die Hauptfunktion von Erklärungen in diesem Kontext?
- Welche Art von Erklärungen kann die Interaktion zwischen Mensch und System in diesem Kontext verbessern?

Im Folgenden wird detaillierter auf die Stakeholderanalyse und der Endnutzeranalyse eingegangen. Anschließend werden Methoden für die Anforderungsanalyse dargestellt. Tjeerd et al. [91] schlagen zusätzlich Methodiken aus der wissenschaftlichen Arbeit von Paetsch et al. [82] vor, welches eine Auflistung von Requirements Engineering Methoden enthält, die ebenso für die Anforderungsanalyse für erklärbare Systeme verwendet werden können.

5.2.4 Stakeholderanalyse

Ein optimales erklärbares System kann alle Erklärbarkeitsziele erfüllen. Der Aufwand dafür darf aber nicht unterschätzt werden. [73] Einige Ziele sind jedoch besonders wichtig für bestimmte Stakeholdergruppen und sollten motiviert durch ihren Bedarf erfüllt werden. Mehrere Arbeiten beschäftigten sich mit der Erfassung von Stakeholdern für erklärbare Systeme. [73] [85] [11] [66] [55] Zusammenfassend werden folgende Stakeholdergruppen genannt: Nicht-technisch versierte Endnutzer, Experten, IT-Experten, Theoristen, Regulatoren, Ethiker, Vorgesetzte und Kunden. Die Erfassung der Stakeholdergruppen ist wichtig, um analysieren zu können, welche Interessen die jeweiligen Stakeholdergruppen an dem erklärbaren System haben. Verwenden sie diese aktiv (nicht-technische Endnutzer, Experten, IT-Experten, etc.)? Fordern sie erklärbare Systeme (Vorgesetzte, Regulatoren, Experten, Ethiker, etc.)? Aus welchen Gründen fordern sie

Endnutzer-typ	**IT Experten-wissen**	**Non-IT Experten-wissen**
Typ 1	X	X
Typ 2	X	✓
Typ 3	✓	X / ✓

Tabelle 5.2: Einordnung der Endnutzertypen bezüglich des zu entwickelnden Softwaresystems. Das X stellt dabei das für die Systemanwendung nicht benötigte oder das nicht vorhandene Wissen dar. Das ✓stellt das vorhandene oder benötigte Wissen dar.

diese (Gesetz-, Ethik-, Marktgetrieben, etc.)? Da die Bedürfnisse und Ziele dieser Gruppen universell sind, müssen diese zunächst erfasst, analysiert und priorisiert werden um letztendlich entscheiden zu können, an wessen Interessen das erklärbare System zugeschnitten sein soll. Hierfür können die RE Methoden, welche im Grundlagenkapitel erwähnt sind verwendet werden. [91] In der Literatur für erklärbare Systeme verwendete Methoden sind in der Tabelle 5.4 zusammengefasst und können ebenso angewendet werden.

Endnutzeranalyse

In dieser Aktivität werden die potenziellen Endnutzer analysiert. Dies ist ein essenzieller Schritt, da Erklärungen nicht universell geeignet sind. Jede Endnutzergruppe benötigt eine an ihn angepasste Erklärung die seine speziellen Wissenslücken schließen soll. Wird der Endnutzer zum Fokus des erklärbaren Systems vereinfacht sich zudem die Problematik der Erklärbarkeit im Ganzen. Dadurch müssen keine universellen Erklärungen erstellt werden, sondern Erklärungen die speziell an die Bedürfnisse aus einer Endnutzergruppe angepasst sind. [77]
In der Literatur werden drei große Endnutzertypen definiert. [73] [85] [11] [21] Eine Zusammenfassung der Endnutzertypen liegt in der Tabelle 5.2 vor. Die Endnutzer können in eine der drei Gruppierungen zugeordnet werden. Alle der drei aufgezählten Endnutzertypen haben unterschiedliche Ziele und Bedürfnisse, die sie mit dem Softwaresystem erfüllen möchten. Die Anforderungen an die Erklärungen ändern sich daher mit der Zuordnung zu einer der Gruppen.
Endnutzer vom Typ 1 verwenden Softwaresysteme unter anderem beruflich oder in ihrem Alltag. Entweder verfügen sie über kein technisches Wissen über die Funktionalität des Softwaresystems oder benötigen ihr Wissen für die Anwendung und Erfüllung ihrer Ziele nicht. Hierrunter fallen Systeme die beispielsweise Video- oder Artikelempfehlungen durchführen.
Endnutzer vom Typ 2 sind Experten aus anderen Fachgebieten wie beispielsweise Humanmediziner. Sie verwenden Erklärungen um das System-

verhalten und die Systemausgabe im medizinischen Kontext zu bewerten und interpretieren zu können. [73] Basierend auf den Ergebnissen und ihrer Interpretation führen sie gegebenenfalls Handlungen aus, wie die Anpassung eines Behandlungsplans eines Patienten.
Endnutzer vom Typ 3 entwickeln Softwaresysteme und verwenden eine Erklärung um das System zu debuggen oder das zugrundeliegende algorithmische Modell zu optimieren und verfügen über ein hohes Maß an Fachwissen. Erklärungen verschaffen ihnen Einblick in die Entscheidungsprozesse des Systems, wodurch Fehler erkannt und behoben werden können. Beispielsweise können Klassifizierungen im maschinellen Lernen nicht relevante Variablen einbeziehen, auf denen der Algorithmus anschließend Entscheidungen trifft, die bezüglich dieser Variablen verzerrt sind. Eine Erklärung darüber welche Variablen in die Entscheidungsfindung miteinfließen kann beispielsweise zur Aufdeckung unvorteilhafter Vorgehen beitragen. [66] Technische Nutzer können gegebenenfalls unterschiedliche Erklärungsmethoden wie z.B. Gradienten verstehen.
Für Endnutzer des Typ 1, sollten diese Erklärungsmethoden nicht verwendet werden, da ein Verständnis nicht vorausgesetzt werden kann. Endnutzer vom Typ 2 können aber von Darstellungs- und Erklärungsmethoden, die üblich für ihr Fach sind, profitieren. Beispielsweise eignet sich in einem Diagnosesystem für einen Arzt der Gebrauch von Fachwörtern und Diagrammen. Für Endnutzer des Typ 1, ist es wichtig von einer natürlichen, leicht verständlichen Sprache Gebrauch zu machen. [57] Bei der Endnutzeranalyse ist daher auch der kulturelle Kontext von Bedeutung. Bei der Entwicklung von Erklärungen sollten möglichst keine mehrdeutigen Ausdrücke oder Symbole verwendet werden. Weiterhin sollte erwähnt werden, dass der Endnutzertyp 3 bei der Verwendung eines Softwaresystems, welches für den Endnutzertyp 1 zugeschnitten wurde, als Endnutzertyp 1 klassifiziert werden kann. Das technische Wissen vom Endnutzertyp 3 ist für die Verwendung des Softwaresystems nicht notwendig und sollte daher nicht zwingend eine Grundlage für Designentscheidungen legen. Geeignete Methoden sind hierfür unter anderem Personas, Szenarios, Interviews, Endnutzerobservierungen oder Fokusgruppen.

Kontext

Weiterhin muss der Kontext der Anwendung analysiert werden. [61] Verwendet der Endnutzertyp 1 das System in seinem Alltag, um seine Zeit zu vertreiben oder lässt er sich eine Autoroute zum Ziel für einen wichtigen Termin bestimmen? Je nach Kontext ändern sich die Anforderungen an das System. Für das erste Beispiel sind fehlerhafte oder störende Erklärungen, wie eine Erklärung die aussagt, dass ein bestimmtes Video dem Endnutzer höchstwahrscheinlich gefallen wird, es im Endeffekt ihm aber doch nicht gefallen hat, sicherlich nicht erwünscht. Sie können gegebenenfalls die

Nutzerzufriedenheit und das Vertrauen in die Vorschläge vermindern, haben aber keine weitreichenden kritischen Auswirkungen. Beim Kontext des Autofahrens sollten jedoch keine fehlerhaften oder störenden Erklärungen erfolgen, da sie unter Umständen den Fahrer verunsichern und dies zu Unfällen führen kann. Im Kontext des Autofahrens würde die Anforderung an die Qualität jeder einzelnen Erklärung viel höher gestellt werden. Ebenso würde die Anforderung an die Art der Darstellung in den beiden Fällen unterschiedlich sein. Während im ersten Beispiel kurze, textuelle Erklärungen den Endnutzer zufriedenstellen würden, stellt das Lesen einer Erklärung während des Autofahrens ein Risiko dar. Hier sollte die Informationsweitergabe beispielsweise per Audio erfolgen, sodass der Fahrer nicht gezwungen wird von der Straße wegzusehen.

5.2.5 Anforderungsinterpretation: Was soll erklärt werden?

Basierend auf den Ergebnissen der Stakeholderanalyse, der Endnutzeranalyse sowie der Kontextanalyse, kann interpretiert werden *Was* der Endnutzer erklärt haben möchte oder ihm erklärt werden sollte. Unterstützend können hierbei wieder endnutzerzentrierte Methoden wie Interviews, Umfragen oder Endnutzerobservierungen verwendet werden. [73] Die Erklärungen sollen in erster Linie den Endnutzer dabei helfen sein übergeordnetes Ziel zu erreichen. Um den Endnutzer nicht mit Informationen zu überfordern, sollten daher nur die Aspekte erklärt werden, die ihm für das Erreichen seines Ziels behilflich sind. Weiterhin haben Liao et al. [67] eine Fragendatenbank für die Entwicklung endnutzerzentrierter erklärbarer Systeme erstellt, die innerhalb der Anforderungsinterpretation unterstützend als Leitlinie verwendet werden kann. Ebenso haben Sokol und Flach [95] in einem *Explainability Fact Sheet* Anforderungen an erklärbare Systeme aus der Literatur gesammelt und definiert. Der Fact Sheet kann zur Evaluierung verwendet werden. [95]Bei der Bestimmung *Was* erklärt werden soll, muss beachtet werden, ob eine Erklärung überhaupt möglich ist.

Ribera et al. [85] zählen die Fragen auf, die laut Lim et al. [68] eine Erklärung beantworten sollte (frei übersetzt aus dem englischen): Was hat das System getan?, Warum hat das System P gemacht?, Warum hat das System nicht X gemacht? Was würde das System tun, wenn Y passiert?, Wie krieg ich das System dazu Z zu machen, im derzeitig gegebenen Kontext? Sie fügen hinzu, dass Gilpin et al. [43] eine zusätzliche Frage definiert: Welche Information beinhaltet das System? Ribera et al. [85] observieren die besondere Wichtigkeit der *Warum* Fragen. Sie schreiben, dass unterschiedliche Endnutzer andere Erklärungsinhalte als wichtig oder interessant einstufen. Auf Basis der vorher erläuterten Aktivitäten können die hierfür benötigten Informationen der Endnutzerbedürfnisse erhoben und anschließend zugeschnittene Erklärungen entworfen werden. Je nachdem wel-

che Fragen der Endnutzer an das System stellt, müssen die Erklärungen auch die passenden Antworten liefern. Tabelle 5.3 listet eine Zusammenfassung der Erklärungsarten und der zugehörigen Fragen auf.

Frage	**Erläuterung**
Wie?	Eine Erklärung darüber, wie das Modell funktioniert
Warum?	Eine Erklärung darüber, warum aus einer Eingabe diese Ausgabe bestimmt wurde. Sie informiert welche Merkmale der Eingabedaten oder welche Logik des Modells zu der Ausgabe geführt hat.
Warum nicht?	Eine Erklärung darüber, warum eine erwartete Ausgabe nicht die tatsächliche Ausgabe ist.
Was wäre wenn?	Eine Erklärung darüber, inwiefern sich die Ausgabe bei einer unterschiedlichen Eingabe verändert.
Inwiefern?	Eine Erklärung darüber, welche Anpassungen der Eingabe oder des Modells zu einem anderen Ergebnis führen würde.
Was noch?	Eine Erklärung darüber, welche anderen Eingaben dieselbe oder ähnliche Ausgaben produziert.

Tabelle 5.3: Eine Auflistung der Fragen die der Endnutzer einem erklärbaren System stellen kann. [73]

Vortrag Für das Co-Design sollte ein Vortrag über die Erklärbarkeit und Problematik für die Teilnehmer gehalten werden, um die Thematik den Teilnehmern näherzubringen. [78] Dies trägt dazu bei, dass die Teilnehmer ein ähnliches Gedankenkonstrukt über die Erklärbarkeit aufbauen können, um auf Basis dessen zu diskutieren und Designansätze zu entwickeln.

Szenarios Das Szenario ist die meist erwähnte Methode in der Literatur. Wolf [104], empfiehlt die Verwendung von Szenarios um erklärbare Systeme zu entwickeln. Sie eignet sich, um die alltäglichen Aufgaben der Stakeholder darzustellen, den Kontext zu erfassen und die Anforderungen an das System zu analysieren. Szenarios regen Spekulationen und Fragen über die Verwendung des Systems in der frühen Entwicklungsphase an, wodurch neue Anforderungen entdeckt werden. [3] Szenarios können auch während einer Endnutzerbefragung unterstützend verwendet werden. Dadurch können sich die Teilnehmer in die Problematik besser hineinversetzen. [78] *Szenarioba-*

siertes Design ermöglicht insbesondere die Analyse des Systems in Bezug zum Kontext während der frühen Entwicklungsphase. Mit dieser Vorgehensweise können sich die Entwickler darauf fokusieren *Was* die Endnutzer verstehen müssen um auf die Systemausgaben angemessen reagieren zu können. Auf folgende Fragen können Antworten gefunden werden: *Welche Art* von Erklärungen könnten Endnutzer bei der Verwendung benötigen? *An wen* richten sich die Erklärungen? [104] Wolf hat insbesondere *Erklärbarkeitsszenarios* definiert, die speziell für die Entwicklung von erklärbaren Systemen verwendet werden sollen. Die erhobenen Anforderungen aus den *Erklärbarkeitsszenarios* bieten eine Grundlage, um anschließend durch Brainstorming technologische Lösungsansätze zu finden. [104]

Persona Eine Persona ist eine archetypische Charakterisierung einer Endnutzergruppe. [58] Sie beschreibt unter anderem das Verhalten, das Ziel, das Bedürfnis und das Problem am Beispiel einer fiktiven Person. Sie hilft den Entwicklern dabei sich in den Endnutzer hineinzuversetzen und ihn zu verstehen. Personas können anhand von qualitativen oder quantitativen Ergebnissen erstellt werden. [60] Sie sind insbesondere für die Definition des Endnutzertyps hilfreich, da unterschiedliche Personas unterschiedliche Anforderungen an die Erklärbarkeit haben. Durch die Klassifizierung und Darstellung der archetypischen Endnutzergruppen können mit Personas auf diese Gruppen zugeschnittene Anforderungen basierend auf ihren spezifischen Bedürfnissen und Zielen erhoben werden. [56] [22] Anschließend kann ein erklärbares System mit diesen Personas auf die Tauglichkeit evaluiert werden. [56] Weiterhin können erklärbare Modelle speziell auf die jeweilige Persona trainiert werden. [5] Dadurch könnten Erklärungen personaspezifisch selektiert und dargestellt werden. [6] beispielsweise ist eine Gruppe des Endnutzertyp 1 an ausführlichen Erklärungen sehr interessiert, eine andere Gruppe legt aber nur auf die absolut notwendigen Erklärungen wert, da sie das Lesen als zu zeitaufwendig einschätzt. Je nachdem wieviel Zeit ein Endnutzer mit einer Erklärung verbringt, könnte dieser in einer der vordefinierten Personas vom System zugeordnet werden, um die Bedürfnisse der Persona zu erfüllen. Hypothethische Personas können anhand von Erfahrungen und der Systemvision definiert werden, um potenzielle Endnutzergruppen zu konkretisieren. Auf Basis dessen können dann reale Endnutzer für weitere Nutzerstudien allokiert werden. [25] Besonders hilfreich sind hypothetische Personas, wenn die Endnutzer für die Anforderungsanalyse nicht zur Verfügung stehen, da sie diese Lücke kompensieren können. Es muss jedoch beachtet werden, dass Personas durch Biases beeinflusst werden können. [60] Ramos et al. [83] haben Personas für erklärbare Systeme aufgestellt, mit dem Augenmerk auf das Vertrauen in Systemausgaben und dem Interesse in die Erklärbarkeit. Sie erhoben die Wahrnehmung und Bedürfnisse der Endnutzer bezüglich der Erklärbarkeit mit einem Fragebogen

und erstellten mit den Daten eine Empathy Map in der festgehalten wurde, was der Endnutzer über Erklärbarkeit sagt, fühlt, tut und denkt. Da die Erstellung von Personas einen gewissen Grad an Kreativität erfordert und dadurch auch die Überprüfung der Validität der Personas sich als schwierig erweist, verwenden die Autoren Empathy Maps um diese negativen Eigenschaften zu umgehen. Ähnliche Empathy Maps wurden zu Personas zusammengefasst, welche anschließend mit einem Fragebogen nach der *Persona Perception Scale* evaluiert wurden. Die *Persona Perception Scale* erfasst Angaben zu der Ähnlichkeit, dem Einfühlungsvermögen, der Sympathie, der Glaubwürdigkeit, der Vollständigkeit und der Klarheit aus der Sicht von Endnutzern und Designern um die Personas auf ihre Angemessenheit zu evaluieren. Mit Personas kann überprüft werden, ob die Anforderungen für den Endnutzertyp erfüllt wurden. [25]

Interview Interviews geben Einsicht in die Wünsche und Bedürfnisse der Endnutzer. Je nachdem welche Fragen gestellt werden, können unter anderem auch Informationen zu Handlungsabläufen, dem Verständniss oder Meinungen erhoben werden. Hall et al. [46] haben semi-strukturierte Interviews für die Erhebung von Anforderungen für erklärbare Systeme verwendet. Dabei merken sie an, dass für ein gegenseitiges Verständnis eine einheitliche Terminologie für jede Rolle verwendet werden muss. Ohne eine vorher definierte Terminologie konnten aus den Interviews keine klaren Anforderungen definiert werden, da jeder zu Kontexten unterschiedliche Wörter verwendete. [46] Beispielsweise werden die Wörter *Erklärbarkeit* und *Verständnis* oft synonymartig verwendet, obwohl das Verständnis nur ein Aspekt der Erklärbarkeit darstellt. Vermeire et al. [100] beschäftigen sich mit der Erfassung von Stakeholderanforderungen für erklärbare Systeme und stellen einen Fragebogen für Interviews bereit. Anhand der erhobenen Antworten können benötigte Eigenschaften der Erklärbarkeit mit den Stakeholderbedürfnissen abgestimmt werden, um so eine geeignete Erklärbarkeitsmethode zu finden.

Umfrage Umfragen und Fragebögen werden häufig verwendet, da sie quantitative, vergleichbare Ergebnisse erheben können. Hierbei wird meist die Likert-Skala verwendet. Oft wird anschließend den Probanden die Möglichkeit gegeben ihre Antwort in einer offenen Frage zu erläutern. Die geschriebenen Texte sind qualitativer Natur und bieten zusätzlich Einsicht über die Entscheidungsbegründung. Ein weiterer Vorteil von Umfragen sind, dass oftmals eine größere Anzahl an Teilnehmern akkredetiert werden können und die Kosten sich im Vergleich zu anderen Methoden eher gering halten. Beispielsweise haben Tsai und Brusilovsky eine online Umfrage mit den bestehenden Endnutzern eines Systems durchgeführt, um den Bedarf unterschiedlicher Erklärbarkeitsziele für ihr Softwaresystem zu bestimmen. [97] In

einer weiteren Arbeit haben sie mit einem Fragebogen Endnutzerpräferenzen erhoben. [98] Weiterhin haben Ramos et al. [83] wie zuvor beschrieben, Endnutzereigenschaften mit einem Fragebogen erhoben, um Personas zu definieren und diese anschließend zu evaluieren.

Endnutzerobservierung Mit einer Endnutzerobservierung können die Bedürfnisse des Endnutzers im jeweiligen Kontext analysiert werden. Beispielsweise können Endnutzer im arbeitsalltag beobachtet werden. Dabei können Informationen über verwendete Tools, Interaktion mit den Tools, Sequenzen der Handlungen, Ziele und Aufgaben, sowie der Unternehmensprozess, in dem der Proband seine Arbeit durchführt, erhoben werden. Die entstehenden Anforderungen und Problematiken können analysiert und von den Entwicklern besser verstanden werden. [92]

Fokusgruppe In einer Fokusgruppe kommen potenzielle Endnutzer zusammen und diskutieren über die Problematik. Dies kann mit anderen Methoden verknüft werden wie einem Szenario und einer kreativen Methodik wie zum Beispiel einer Zeichnung. Durch eine Fokusgruppe können Diskussionen über unterschiedlichste Endnutzeranforderungen, wie unter anderem, die Bedürfnisse, Ziele und Meinungen angeregt werden. Anhand dessen entsteht auch das Potenzial innovativer Ideenfindung für erklärbare Systeme, insbesondere wenn kreative Aufgaben involviert werden. [78]

5.2.6 Back-End Analyse

Unabhängig von den zuvor gelisteten Methoden sollte die Aktivität der Back-End Analyse dennoch nicht unerwähnt bleiben. In dieser wird die algorithmische Funktionsweise des Softwaresystems analysiert oder geplant. Abhängig von den Stakeholderanforderungen muss entschieden werden, ob eine lokale oder globale Erklärbarkeit integriert werden soll. Wenn die Anforderung an einer globalen Erklärbarkeit des Systems existiert, muss in dieser Aktivität geprüft werden, ob eine globale Erklärbarkeit des Systems möglich ist und wie diese umgesetzt werden kann. [9] Oftmals beinhaltet diese Entscheidung für eine globale Erklärbarkeit eine negative Abhängigkeit mit der Genauigkeit der Systemberechnungen. Dieser *Trade-off* der Anforderungen untereinander wird im Folgenden näher erläutert.

Trade-off Deep Learning Modelle können genaue Systemausgaben erzeugen. Gleichzeitig stellen sie eine Black Box dar, die nicht direkt erklärbar ist. Selbst die Entwickler des Models können oft die Entstehung der Ergebnisse kaum nachvollziehen. Hierduch entsteht ein Trade-off in Bezug auf die Erklärbarkeit und der Ergebnisgenauigkeit. Daher muss mit den beteiligten Stakeholdern abgestimmt werden, ob die Genauigkeit höher oder niedriger priorisert wird als die Erklärbarkeit des Systems. [88] Die Grafik 5.2 stellt

die gängigen ML-Modelle im Trade-off der Präzision und Erklärbarkeit dar. Die Entscheidung, welches Modell letztendlich verwendet wird, ist stark kontextabhängig. [9] Eine Erklärung kann aber auch von einem zusätzlichen Modell generiert werden, wenn sie nicht schon selbsterklärend ist, wie z.B. kleine Entscheidungsbäume.

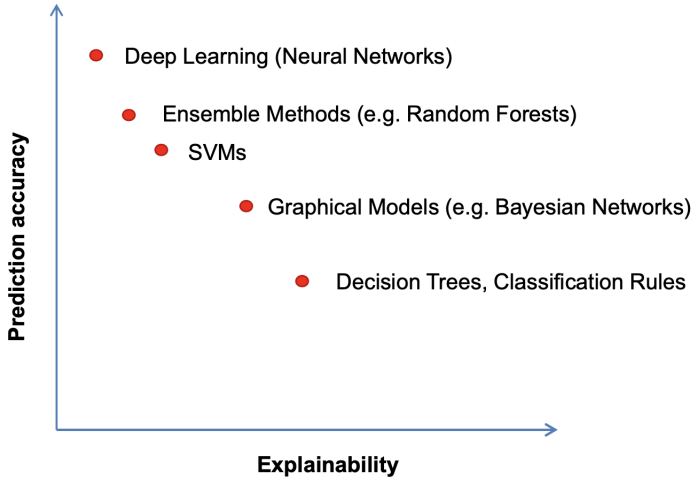

Abbildung 5.2: Der Trade-off von Präzision und Erklärbarkeit. Entnommen aus [32].

Die globale Erklärbarkeit ist hauptsächlich für Endnutzer des Typ 3 relevant. Diese können aber auf Basis des globalen Wissens Erklärungen für die Endnutzertypen 1 und 2 erstellen. Ein anderer Ansatz können post-hoc Erklärungen sein, bei der ein zusätzliches Modell entwickelt wird, welches dabei hilft, die Entscheidungen des Black Box Modells zu interpretieren oder vorauszusagen. [107] Weiterhin kann ein erklärbares System auf lokaler Ebene erstellt werden, in der die Beziehung von Eingabe und Ausgabe erklärt wird. [50] Diese Art der Ergebnisbegründung ist für alle Endnutzertypen geeignet. Da die Entwickler das Back-End erstellen, verfügen sie über das Expertenwissen des Back-Ends und können das *mentale Modell des Experten* aufstellen, welches die korrekte Funktionsweise des Softwaresystems darstellt und für die Evaluationsaktivität verwendet werden kann. [38] Es werden aber nicht immer komplexe Modelle im Back-End eines Softwaresystems verwendet. Auch ein System ohne künstliche Intelligenz kann erklärbar gestaltet werden. Bei diesen Systemen ist eine Back-End Analyse nicht zwingend erforderlich. Abbildung 5.3 stellt die Anforderungsabstimmung grafisch dar.

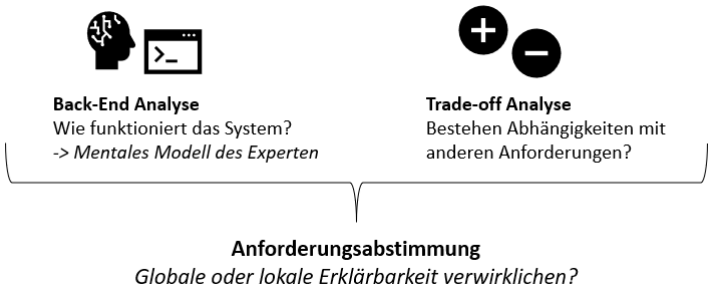

Abbildung 5.3: Workflow für die Anforderungsabstimmung

5.2.7 Dokumentation

Oftmals werden nicht-funktionale Anforderungen nicht dokumentiert oder nur schlecht strukturiert, vage und mit nicht messbaren Konditionen definiert. [2] Nguyen et al. [79] dokumentieren die Erklärbarkeitsanforderungen in einem *Explainability Requirement Template*. In diesem Template wird der Prozess, die Quelle der Anforderung, der betroffene Stakeholder, der Grund für die Notwendigkeit der Erklärung und die Priorität dokumentiert. Dadurch wird anschließend der Fokus der Erklärbarkeit abgeleitet. Die Abbildung 5.4 stellt die *ExplainabilityList* und ihren Prozess grafisch dar. Köhl et al. [61] haben eine Textschablone erstellt, um Erklärbarkeitsanforderungen nach vorgegebenem Schemata festzuhalten (aus dem englischen frei übersetzt):

> *Ein System S muss erklärbar für die Zielgruppe Z im Kontext K mit Anbetracht auf Aspekt Y des Explanadum X sein.*

Wobei mit *Explanadum* jegliche Softwareausgabe gemeint wird, die erklärt werden sollte.

Weiterhin wird empfohlen NfA-Kataloge und SIGs zu verwenden. [25] Sokol und Flach [95] haben ein *Explainability Fact Sheet* zusammengestellt, in der alle erhobenen Anforderungen und die Art der Erfüllung dieser festgehalten werden sollen.

Use Case Wann was dem Endnutzer erklärt werden soll, könnte in Use-Cases festgehalten werden. Die schrittweise Beschreibung des Systemverhaltens eignet sich gut dafür den Zeitpunkt und die Motivation, warum die Erklärung gerade benötigt wird, festzuhalten. Zum Beispiel ändert ein Navigationssystem die berechnete Route augrund eines Staus. Dann würde in einem nicht-erklärbaren System der Endnutzer über diese Änderung nicht informiert werden. In einem erklärbaren System wäre der nächste Handlungsschritt im Use Case: *Endnutzer mit einer Erklärung informieren.*

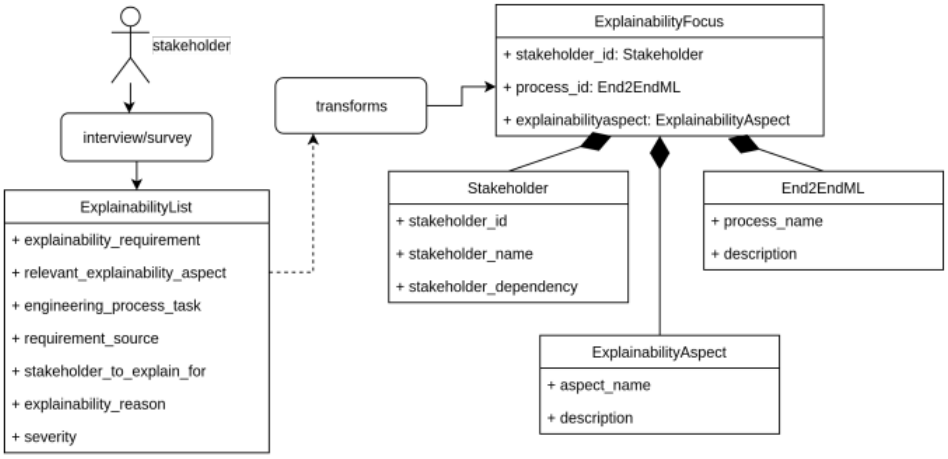

Abbildung 5.4: Darstellung des Vorgehens bezüglich des Requirements Engineerings entnommen von Nguyen et al. [79].

5.2.8 Verifikation und Validation

In der Verifikation und Validation können Walkthroughs, wie Expertenevaluationen durchgeführt werden. [39] Diese Vorgehensweise ist für Softwaresysteme für den Endnutzertyp 2 geeignet. Tjeerd et al. [91] haben beispielsweise ein erklärbares System für Humanmediziner entwickelt. Das System wurde von einem Fachexperten bezüglich ihrer Erklärbarkeit verifiziert und validiert. Dabei ging dieser auf das Aspekt, wann welche Erklärung wichtig ist, ein. Dieser Experte war zuvor auch bei der Definition der Use Cases involviert, wodurch er Anteilnahme an der Anforderungsdefinition hatte. Weiterhin haben sie einen Fragebogen entworfen mit dem sie die Relevanz der Erklärungen durch weitere Humanmediziner validieren. [91] Köhl et al. [61] empfehlen die Erklärbarkeit mit dem Verständnis der Stakeholder zu verknüpfen, da die Erklärbarkeit kein technisches Konzept ist. Dadurch wird eine Verifikation der Erklärbarkeit ermöglicht, indem das Verständnis der jeweiligen Stakeholder durch Evaluationsmethoden gemessen wird. Sokol und Flach [95] empfehlen, dass die Validation ebenso durch Evaluationsmethoden in einer ähnlichen Umgebung des Anwendungskontextes erfolgen soll (siehe dazu Sektion 6). Dies ist wichtig, das der Kontext großen Einfluss auf die Erklärbarkeitsanforderungen hat.

5.2.9 Anforderungsmanagement

Im Anforderungsmanagement sollen Veränderungen nachvollziehbar dokumentiert werden und rückverfolgbar sein. Um dies für die Erklärbarkeit zu realisieren, kann die *ExplainabilityList* verwendet werden, die im Laufe der Entwicklung kontinuierlich aktualisiert werden soll. [79] Ebenso kann die

Methode	Quelle
Szenario	[104] [3] [30] [78] [40] [55] [66] [96] [57] [5] [36] [37] [10] [106] [85] [75] [76] [84] [81]
Persona	[25] [83] [3] [96] [8] [106] [22] [56] [21]
Interview	[46] [52] [100] [57] [99] [107] [73] [38] [90] [97] [35] [44] [20] [30]
Umfrage	[83] [100] [57] [76] [98] [73] [38]
Nutzerobservierung	[92]
Fokusgruppe	[78] [57]
Empathie Modell	[83] [108]
Card Sorting	[57] [38]
Workshop	[107]
Vortrag	[78]

Tabelle 5.4: Literaturergebnisse für die Anforderungsanalyse

Explainability Fact Sheet von Sokol und Flach [95] für die Versionierung und Rückverfolgbarkeit verwendet werden. Der Fact Sheet soll laut den Autoren bei Aktualisierungen systematisch versioniert werden, wodurch auch eine Rückverfolgung der Änderungen möglich wird. Weiterhin konnte innerhalb des Literaturreviews keine Methodik und auch keine weiteren Aktivitäten bezüglich des Anforderungsmanagements gefunden werden. Insbesondere durch die kontinuierliche Aktualisierung der Designentscheidungen ist das Anforderungsmanagement ein wichtiger Faktor für die Entwicklung erklärbarer Systeme. Der Grund der Anpassung einer Erklärbarkeitsanforderung sollte nicht im Laufe des Entwicklungsprozesses vergessen oder verloren werden.

5.3 Softwareentwurf

Im Softwareentwurf wird die Softwarestruktur geplant und entworfen. Hier wird eine Antwort auf die Frage *Wie* erklärt werden soll gesucht.

5.3.1 Entwurf

Im Entwurf muss entschieden werden in welcher Art und Weise die Erklärung mit dem Endnutzer kommuniziert wird, sodass die Endnutzererfahrung positiv ist. Die Entscheidungen hierzu können beispielsweise in einem *Erklärbarkeitsguideline* festgehalten werden. Je nach Kontext und Endnutzertyp müssen angepasste Designentscheidungen getroffen werden, die in der

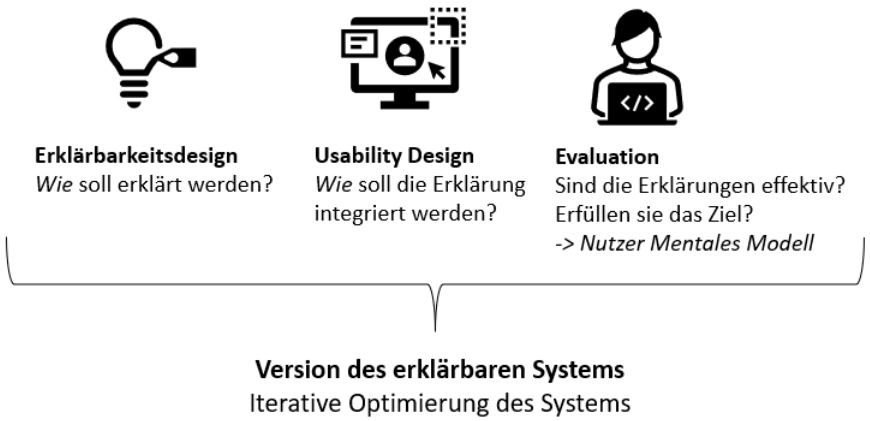

Abbildung 5.5: Workflow für Designentscheidungen und Evaluation

Evaluationsaktivität auf Eignung getestet werden sollten.

Statisch oder interaktiv Erklärungen können dem Endnutzer statisch dargestellt werden oder interaktiv gestaltet sein. Eine interaktive Erklärung kann beispielweise ein Chatroboter sein, dem der Endnutzer seine Fragen stellt und das System ihm antwortet. [99] Auch interaktive Grafiken können erklärend sein, wenn beispielsweise bei der Auswahl eines Elements eine genauere Erklärung hierfür dargestellt wird. Da interaktive Erklärungen dem schrittweisen Kommunikationsmuster von Menschen ähnlicher sind, fördern sie das objektive Verständnis des Endnutzers eher, im Vergleich zu statischen Erklärungen. [26]

Formulierung Natürliche Sprache kann für alle drei Gruppen verwendet werden, das Vokabular kann sich aber bzgl. des Wissensstands unterscheiden. Erklärungen für Endnutzer sollten keine Fachwörter enthalten, während die Verwendung von Fachwörtern für Domain-Experten vorteilhaft ist. Daher sollte hierbei insbesondere auf die typische Formulierungsart der Endnutzergruppen geachtet werden. Innerhalb des Endnutzertyp 1 fallen beispielsweise jüngere, sowie auch ältere Generationen zusammen. Anhand einer Persona sollten hier die Differenzen der Personas in den Designentscheidungen berücksichtigt werden.

Zeitpunkt Rosenfeld et al. [88] betrachten auch den Zeitpunkt der Darstellung einer Erklärung. Demnach kann eine Erklärung vor der Aufgabe erfolgen, wie beispielsweise eine Anleitung. Sie kann während der Anwendung der Software geschehen, wie eine Navigationssoftware dies während einer Autofahrt tut oder aber nach der Aufgabe um beispielsweise Systemausgaben zu erklären.

Informationsdarstellung Das Medium, wie die Erklärung dem Endnutzer zur Verfügung gestellt wird, muss bestimmt werden. Eine Erklärung kann unter anderem visuell, textuell, auditiv, per Haptik oder einer Mischung derer erfolgen. Für Expertensysteme eignen sich auch analytische Ansätze die auf Zahlen basieren. Darüber hinaus wird auch argumentiert, dass Interaktivität über das Interface für die Erklärbarkeit beiträgt, da es den Prozess für den Endnutzer natürlicher gestaltet.

Prototyp

Für erklärbare Systeme sind wirkungsvolle Erklärungen ein potenziell entscheidender Aspekt für ihre Nützlichkeit und Verwendbarkeit. Deshalb ist die Art und Weise wie die Informationsvermittlung durch Erklärungen für den Endnutzer dargestellt wird von besonderer Bedeutung. [13] Erklärungen müssen für den Endnutzer verständlich oder nachvollziehbar sein, ihre offenen Fragen beantworten oder bei der Durchführung ihrer Aufgaben unterstützen. [4] Prototypen eignen sich um Designideen und unterschiedliche Darstellungsformen von Erklärungen in visuell ansprechender und effektiver Weise darzustellen. Sie können auf ihre Wirksamkeit und die Erfüllung der Endnutzerbedürfnisse aus der Anforderungsanalyse evaluiert werden, weshalb Prototypen auch für die Entwicklung von erklärbaren Systemen oft erstellt werden (siehe Tabelle 5.5). [4]

Low-Fidelity Prototyp Ein *Low-fidelity Prototyp* stellt eine unkomplizierte, schnelle und kostengünstige Methode dar, um Ideen und Designentscheidungen, bevor sie implementiert werden, in einem Trial-and-Error Verfahren auszuprobieren und mit Endnutzern zu testen. [57] Beispielsweise können, je nach Kontext, *Papier-Prototypen, Mock-Ups* und *Wireframes* gut geeignet sein. Oft wird in Kombination die *Think-Aloud* Methode und ein *Walkthrough* angewendet. [52] [78]

Papier Prototyp *Papier Prototypen* können für den Entwurf, aber auch zur Evaluation von Designentscheidungen verwendet werden. Die Endnutzer oder Entwickler können sie in jeglicher Art und Weise an ihre Bedürfnisse anpassen, ohne vorerst die Funktionsweise eines Zeichentools erlernen zu müssen. Gleichzeitig werden auch die einschränkenden Ausdrucksmöglichkeiten von Softwarezeichentools umgangen. [65]

Mock-Up Mit Mock-Ups können Designansätze visualisiert werden und das *Brainstorming* für weitere Ansätze kann angeregt werden. Sie stellen die Oberfläche des Systems dar, womit auch eine Endnutzerevaluation durchgeführt werden kann, um die Nutzererfahrung zu simulieren. [78] Mock-Ups können auch *Medium-fidelity Prototypen* sein, indem die Funktion anhand von Tools simuliert werden kann. Dem Endnutzer können die

Funktionen der Software in der Oberfläche dargestellt und von ihm bedient werden. Dadurch kann nachvollzogen werden, wie der Endnutzer das System verwendet. [4] [52] Beispielsweise haben Kallina et al. [59] ein Mock-Up erstellt, um ihre Hypothese bezüglich des Softwareentwurfs zu testen.

High-fidelity Prototyp

Durch die Evaluation mit *High-fidelity Prototypen* können präzisere Ergebnisse erhoben werden, da die Endnutzer mit dem (nahezu) tatsächlichen System interagieren können. Die Entwicklung eines High-fidelity Prototyps mit realem Datensatz und komplettem Algorithmus ist jedoch zeitaufwändig und kostspielig. [48] Die vollständige Implementierung des Systems ist aber nicht zwingend erforderlich. Beispielsweise können manuell generierte Daten zum Testen eines vollständig implementierten Front-Ends verwendet werden, um das Systemdesign testen zu können, ohne dass die zeitintensive Implementierung des Back-Ends erfolgen muss. [105] Ebenso können mit der *Wizard-of-Oz* Methodik die Berechnungen im Back-End durch eine Person vorgespielt werden. Dies umgeht das Problem innerhalb der Entwicklung von Systemen mit einer künstlichen Intelligenz, da Prototypen meist auf keinem realen, vollständigen oder komplexen Datensatz oder Algorithmus basieren. [48]

Methode	Quelle
(High-fidelity) Prototyp	[97] [48] [26] [108] [105] [38]
Low-fidelity Prototyp	[57] [96] [108] [105] [38]
Papier Prototyp	[65]
Mock-Up	[78] [47] [52] [59] [73] [19] [91] [27] [105]

Tabelle 5.5: Literaturergebnisse für das Design

5.3.2 Kodierung

Da sich diese Masterarbeit mit der endnutzerzentrierten Entwicklung von erklärbaren Systemen beschäftigt, werden unterschiedliche algorithmische Verfahren für erklärbare Systeme nicht genauer erläutert. Methoden aus dem RE oder MMK die sich für die Kodierung eignen könnten, wurden in den untersuchten wissenschaftlichen Texten nicht erwähnt. Es ist nicht auszuschließen, dass Methoden zwar verwendet wurden, aber nicht explizit erwähnt wurden. In dieser Aktivität erfolgt daher die reine Implementierung der Designentscheidungen, beispielsweise durch die Erstellung eines *High-fidelity Prototyps*

5.4 Evaluation

In der Evaluationaktivität wird überprüft, ob das System erklärbar ist. Der Fokus bei dieser Aktivität liegt auf dem Endnutzerfeedback in Kombination mit einem Prototyp, da die Wirksamkeit und Qualität einer Erklärung subjektiv ist. Ein System auf seine Erklärbarkeit zu evaluieren ist insbesondere deshalb auch eine herausfordernde Aufgabe. [70] Da jeder einzelne aus einer repräsentativen Gruppe andere kognitive Prozesse beim Verständnis einer Thematik hat, müssen tiefergehende Analysen des Verständnisses vorgenommen werden. Hierbei eignen sich Methoden aus der Bildungs- und Kognitionswissenschaft, welche sich mit der Thematik des menschlichen Verständnisses ausgiebig beschäftigen. [61] Die Güte einer Erklärung basiert auf dem Anwendungskontext und den Zielen des Endnutzers. [77] Erklärungen müssen an die Interessen der Endnutzergruppe angepasst werden. Je nachdem welcher Endnutzertyp welches Ziel mit dem erklärbaren System erfüllen möchte, ändern sich die geeigneten Evaluationsmetriken. [73] Für den Endnutzertyp 1 sind beispielsweise wichtige Metriken ein korrektes mentales Model, die Nützlichkeit des Systems und die Zufriedenheit während und nach der Verwendung. [29] Eher nicht ausschlaggebend für den Endnutzertyp 1 ist das Ziel des Debuggings, dafür aber besonders wichtig für den Endnutzertyp 3. Je nach Endnutzertyp können unterschiedliche Verfahren angewendet werden, um sie zu rekrutieren. Für den Endnutzertyp 1 eignet sich das Crowdsourcing, um eine hohe Anzahl an Teilnehmern zu finden. Probanden der Endnutzertypen 2 und 3 sind üblicherweise schwerer zu finden. Für diese Endnutzergruppen eignen sich Onlinestudien, Laborstudien oder Feldstudien. [29] Für die Evaluation von Erklärungen oder eines erklärbaren Systems ist ein Experiment mit einem Prototyp essenziell, damit folgenden Evaluationsmethoden angewendet werden können.

5.4.1 Mentales Modell

Da Erklärungen Wissenslücken füllen, ist der kognitive Prozess des Endnutzers essenziell. Das *mentale Modell* bietet eine gute Möglichkeit diesen Prozess zu erfassen. [38] Entstanden aus der Psychologie und von der MMK adaptiert, werden mentale Modelle verwendet, um das Verständnis des Nutzers im Bezug auf das Softwaresystem zu erfassen. [73]Um nachvollziehen zu können, wann eine Erklärung benötigt wird, muss verstanden werden in welchen Bereichen es an Verständnis mangelt. Um ein konzeptionelles Verständnis eines Endnutzers zu gewährleisten, wird in der Evaluationsaktivität ein mentales Modell erstellt. Das mentale Modell stellt das Verständnis des Endnutzers vom Softwaresystem dar, mit allen Vorurteilen oder Erwartungen die der Endnutzer gegenüber dem System hat. Wenn eine Situation eintritt, die nicht vom mentalen Modell des Endnutzers abgedeckt ist, benötigt er eine Erklärung um den entsprechenden Bereich

in seinem mentalen Model zu korrigieren.Fehler- oder lückenhafte mentale Modelle des Endnutzers müssen mit Erklärungen korrigiert werden. Eine Verbesserung des mentalen Modells trägt für die Benutzerzufriedenheit, dem Vertrauen und der wahrgenommenen Kontrolle in das System und seine Ausgaben positiv bei. Erklärungen über die Funktionsweise des Systems verbessern nachweislich die mentalen Modelle der Endnutzer. Die Diskrepanz zwischen dem mentalen Modell und der tatsächlichen Funktionsweise des Softwaresystems stellt den Erklärungsbedarf des Endnutzers dar, welcher mit einer Erklärung geschlossen werden soll. [73] Um das Modell evaluieren zu können, wird ein mentales Modell des Endnutzers und ein optimales mentales Modell von einem Experten in der *Back-End Analyse* erstellt. Beide Modelle werden anschließend miteinander verglichen. Abweichungen des Endnutzermodells vom Expertenmodell zeigen Missverständnisse auf, die es im weiteren Entwicklungsprozess zu beheben gilt. [38] Ein mentales Modell kann mit Ergebnissen aus *Interviews, Think-Alouds, Selbsterklärungen, Fragebögen* und *Output- oder Fehlervorhersagen* erstellt werden. [73] Hoffman et al. [51] schreiben, dass zehn bis zwölf Endnutzer für die Erstellung eines repräsentativen mentalen Modells ausreichen. Für die Bewertung mentaler Modelle sollte mehr als eine Methode eingesetzt werden. Die Leistung bei einer bestimmten Art von Aufgabe stimmt möglicherweise nicht mit der Leistung in einer anderen Aufgabenart überein. So führten Hoffman et al. [51] Evaluierungen eines erklärbaren Systems durch, bei der ein Endnutzer die Funktionsweise in einem Diagramm korrekt darstellte, aber bei einer Outputvorhersage keine optimale Leistung erbrachte. Die Möglichkeit, dass die Leistung der Endnutzer innerhalb einer Methode gut sein kann und dennoch das mentale Modell fehlerhaft oder unvollständig ist, ist gegeben. Menschen sind nicht in der Lage ihr Verständnis so genau zu formulieren, dass ein Zweiter es vollkommen nachvollziehen kann. Mit einer angemessenen Unterstützung durch eine Methode wird dies den Endnutzern aber erleichtert. So können sie ihr Verständnis kontextbezogen und dadurch spezifischer darstellen. [51] Im Folgenden werden in der Literatur verwendete Methoden für die Evaluation vorgestellt.

Interview Mit Interviews kann qualitatives Feedback der Endnutzer eingeholt werden. Beispielsweise kann nach der Verwendung eines Prototyps die Meinung bezüglich des Softwaresystems erfragt werden. Hierbei kann speziell die Nützlichkeit, die Zufriedenheit und das Vertrauen in das erklärbare System thematisiert werden. [28] Mehrere Versionen des Systems können dem Probanden vorgestellt werden, welcher diese vergleicht und den Grund seiner Präferenz mitteilt. [97]

Selbsterklärung Ob ein Endnutzer eine Erklärung verstanden hat kann durch eine Selbsteinschätzung abgefragt werden. Leider birgt diese Methode

einige Problematiken. Kognitive Voreingenommenheiten, wie eine Selbstüberschätzung oder -unterschätzung, können das Ergebnis verfälschen. [61] Bei dieser Methode ist daher eine Verknüpfung zu einer Aufgabe nützlich. So können dem Endnutzer Selbsterklärungsaufgaben gestellt werden, anhand dessen das eigene Verständnis zum Ausdruck gebracht werden kann. [51] Hier besteht eine Vielzahl an Möglichkeiten die Aufgaben zu gestalten.
Ein Ansatz wäre die Beantwortung der eigenen Fragen mithilfe der Erklärungen. Mit einem anschließenden *Wissenstransfer* kann das Verständnis nocheinmal bestätigt werden. [61] Bei der *Nearest Neighbor Task*, wählen die Endnutzer eine Erklärung oder ein Diagramm aus, welches nach ihrem Verständnis das System am besten beschreibt.
Die *ShadowBox Lite* Aufgabe ist insbesondere im XAI-Kontext anwendbar. Hierbei wird keine ausführliche Selbsterklärung des Endnutzers benötigt, um das mentale Modell zu analysieren. Es wird eine Frage zu der Funktionsweise der Software gestellt, der ein Erklärungsvorschlag beigefügt ist. Der Endnutzer soll dann eine oder mehrere Möglichkeiten auswählen, in denen der Erklärungsvorschlag eher gut oder eher schlechter geeignet ist. Ein Experte ordnet vorher ebenso die Erklärungsvorschläge nach guter und schlechter Eignung zu. Dem Endnutzer wird dann der Vergleich der Ergebnisse dargestellt. Dadurch können neue Erkenntnisse erlangt werden. [51]
Bei der *Diagrammaufgabe*, soll der Endnutzer ein Diagramm erstellen, das ihr Verständnis über das System darstellt. Die Entwickler können anhand des Diagramms Fehler im mentalen Modell des Endnutzers finden, sowie die mentalen Modelle unterschiedlicher Endnutzer vergleichen. [51]
Bei der *Aufgabenreflexion* sollen die Endnutzer ihre Gedanken nach der Bearbeitung einer Aufgabe mit dem Softwaresystem beschreiben. Dafür kann der Bearbeitungsprozess auch in einem Video festgehalten werden. Diese Aufnahme kann dem Endnutzer im Anschluss gezeigt werden, der sein Vorgehen dann reflektiert. [51] Anhand dessen können Erkentnisse gewonnen werden, warum der Endnutzer an gewissen Zeitpunkten so gehandelt hat.

Ausgabevorhersage Auch Vorhersageaufgaben eignen sich zur Evaluierung erklärbarer Systeme. Die Endnutzer werden hierzu aufgefordert die Ergebnisse des Softwaresystems vorauszusagen. Dabei handelt es sich um eine einfache Methode, um mentale Modelle zu untersuchen. Beispielhafte Fragen können sein: *Was wird als nächstes geschehen?* oder *Warum sollte nicht etwas anderes geschehen?* Anschließend sollen sie erläutern, warum sie dachten, dass das vorhergesagte Ergebnis eintreten würde, um den Grund der Annahme zu durchleuchten. [51]

Think-Aloud Um mentale Modelle zu erstellen, eignet sich die Think-Aloud Methode in Kombination mit einer Aufgabenstellung und einem Prototyp, in der die Endnutzer ihre Gedankengänge verbalisieren. Während

der Aufgabenbearbeitung können dem Endnutzer auch konkrete Fragen gestellt bezüglich seines Verständnisses gestellt werden. [51]

Szenario Ein Szenario kann genutzt werden, um dem Endnutzer den Kontext der Aufgabe zu vermitteln, inwelches das Softwaresystem verwendung findet. Negativbeispiele können die Wichtigkeit der richtigen Lösung verdeutlichen. Herm et al. [49] haben beispielsweise den Teilnehmern ein Szenario und zugehörige Informationen wie den Datensatz, das Aufgabenziel und die Auswirkungen von Fehlentscheidungen dargestellt. Bei einem Verwendungsszenario wird die fiktive Vorgehensweise einer fiktiven Person bei der Verwendung des entwickelten Systems sehr detailliert beschrieben. Dabei wird unter anderem auf die Motivation, Aufgabe, Klicks, Informationswahrnehmung der Persona, Aktivitäten und Gedanken der Persona eingegangen. Diese Methode eignet sich, falls keine Endnutzer für die Evaluationsphase verfügbar sind.

Quiz Ein Quiz kann verwendet werden, um das objektive Verständnis des Endnutzers über das System zu erheben. Beispielsweise können Fragen mit oder ohne Antwortmöglichkeiten gegeben werden, in der der Endnutzer beantworten soll, welche Attribute die Systemausgabe beeinflussen oder welche Vorhersage der Systemausgabe bei Änderungen der Eingabe korrekt ist. [26]

Expertenevaluation Insbesondere für Softwaresysteme die für den Endnutzertyp 2 entwickelt werden, können von Expertenevaluationen profitieren. Chari et al. [23] haben beispielsweise im Bereich der Humanmedizin ein erklärbares System entwickelt, welches mit einem Experten evaluiert wurde. Der Experte hob positive und negative Aspekte des Softwaresystems hervor und hat Möglichkeiten zur Verbesserung vorgeschlagen. Der Experte sollte ein potenzieller Endnutzer mit der benötigten Praxiserfahrung sein, um den Kontext der Systemanwendung einschätzen zu können.

Zeitmessung Eine quantitative Evaluierungsmöglichkeit bietet die Messung der Zeit während der Verwendung eines erklärbaren Systems. [26] Damit kann erfasst werden wie effizient das System mit den Erklärungen bedient werden kann. Dauert beispielsweise das Lesen der Erklärungen zu lange, könnte die Endnutzererfahrung darunter leiden und das System ist möglicherweise ineffizient.

Fragebogen Ein Likert-Skala Fragebogen kann durch Selbstangaben der Endnuzer zur Erfassung des Verständnisses beitragen. Cheng et al. [26] haben hierfür sieben Skalen verwendet. Sie stellten die Frage *Ich verstehe den Aufnahemealgorithmus* mit der Antwortspanne von *Starke Ablehnung* bis

hin zu *Starke Befürwortung*. Weiterhin erheben sie mit dem Fragebogen die zeitliche Dauer, das Vertrauen, die technische Versiertheit, die algorithmische Versiertheit, demografische Informationen der Teilnehmer und mit offenen Fragen das *Warum*. [26] Nützlichkeit, Zufriedenheit und Vertrauen in erklärbare Systeme können mit Fragebögen erhoben werden. [28]

Ein Fragebogen zur Evaluation kann in Zusammenhang mit einem zuvor gehaltenen Experiment erfolgen. Dadurch können spezielle Fragen bezüglich der Effektivität der Erklärbarkeit im Bezug auf das Softwaresystem gestellt werden. [76] Hoffman et al. [51] haben ausführliche Fragebögen zur Evaluierung der Qualität von Erklärungen aufgestellt und inwiefern sie bestimmte Eigenschaften der Erklärbarkeit erfüllen. Diese Fragebögen umfassen die Güte der Erklärung, ob die Endnutzer mit den Erklärungen zufrieden sind, wie gut die Endnutzer die KI-Systeme verstehen, wie die Neugierde die Suche nach Erklärungen motiviert, ob das Vertrauen des Endnutzers in die KI angemessen ist und wie das Softwaresystem funktioniert. Selbst wenn eine Erklärung "gut" sei, bedeutet das nicht automatisch, dass sie für die Endnutzer auch zufriedenstellend ist. Deshalb wurde für die Güte der Erklärung eine *Explanation Goodness Checklist* aufgestellt, mit der die Qualität von Erklärungen durch die Endnutzer bewertet werden können. Dabei wird die Effektivität einzelner Eigenschaften mit einer *Ja* oder *Nein* Antwortmöglichkeit abgefragt. Folgende Fragen wurden definiert, die frei aus dem englischen übersetzt sind [51]:

- Die Erklärung hilft mir zu verstehen, wie die [Software, der Algorithmus, das Tool] funktioniert.
- Die Erklärung der Funktionsweise der [Software, des Algorithmus, des Tools] ist zufriedenstellend.
- Die Erklärung der [Software, des Algorithmus, des Werkzeugs] ist ausreichend detailliert.
- Die Erläuterung der Funktionsweise der [Software, des Algorithmus, des Werkzeugs] ist ausreichend vollständig.
- Die Erklärung ist umsetzbar, d. h. sie hilft mir zu wissen, wie ich die [Software, den Algorithmus, das Tool] verwenden kann.
- Die Erklärung lässt mich wissen, wie genau oder zuverlässig die [Software, der Algorithmus] ist.
- Die Erklärung lässt mich wissen, wie vertrauenswürdig die [Software, der Algorithmus, das Werkzeug] ist.

Weiterhin stellen Hoffman et al. [51] eine *Explanation Satisfaction Scale* auf, mit der die Zufriedenheit der Endnutzer in Bezug auf die Erklärung in einem separaten Fragebogen befragt werden. Hier wird eine fünf-stufige

Likert-Skala verwendet um die Endnutzer zu einzelnen Eigenschaften einer Erklärung zu befragen.

System Causability Scale (SCS) Das System Causability Scale ist inspiriert vom System Usability Scale. Sie wird verwendet um die Effektivität von erklärbaren Oberflächen, Erklärungen oder eines erklärenden Prozesses in einer quantitativen Art zu bewerten. Umgesetzt wird dies mit einem Fragebogen aus zehn Fragen mit Antwortkategorien der Likert Skala (aus dem englischen frei übersetzt). [53]:

- Ich habe festgestellt, dass die Daten alle relevanten bekannten Kausalfaktoren mit ausreichender Präzision und Granularität enthalten.
- Ich habe die Erklärungen im Zusammenhang mit meiner Arbeit verstanden.
- Ich konnte den Detaillierungsgrad bei Bedarf ändern.
- Ich brauchte keine Unterstützung, um die Erklärungen zu verstehen.
- Ich fand, dass die Erklärungen mir geholfen haben, die Kausalität zu verstehen.
- Ich war in der Lage, die Erklärungen mit meiner Wissensbasis zu nutzen.
- Ich habe keine Unstimmigkeiten zwischen den Erklärungen gefunden.
- Ich denke, dass die meisten Menschen die Erklärungen sehr schnell verstehen.
- Ich habe keine weiteren Verweise in den Erklärungen benötigt: z.B. medizinische Richtlinien, Vorschriften.
- Ich habe die Erklärungen zeitnah und effizient erhalten.

Die Nachteile der Likert Skalen Methode müssen bei der Auswertung des SCS beachtet werden, da ansonsten falsche Schlussfolgerungen gezogen werden könnten (siehe hierfür [97])

Versionen vergleichen Mehrer Versionen eines Softwaresystems können von Endnutzern evaluiert werden, um die optimale Version zu finden. Beispielsweise kann das System mit Erklärungen gegenüber dem System ohne Erklärungen evaluiert werden oder detaillierte Erklärungen mit gefilterten, weniger detaillierte Erklärungen. [76] Auch können Erklärungsarten miteinander verglichen werden und vom Endnutzer bewertet werden. [49] [29]

Methode	Quelle
Mentales Modell	[97] [73] [38] [65] [99] [51]
Experiment mit Prototyp	[76] [98] [108] [38]
Interview	[97] [73] [78] [19] [99] [84] [108] [38]
Selbsterklärung	[51] [61]
Ausgabevorhersage	[51] [80]
Think-Aloud	[78] [108] [38]
Szenario	[78] [27] [49] [38]
Quiz	[26] [84] [108] [81]
Expertenevaluation	[23]
Zeitmessung	[26]
Fragebogen	[26] [75] [76] [98] [49] [64] [81] [38]
System Causability Scale (SCS)	[53]
Versionen vergleichen	[76] [49] [62]
Card Sorting	[97] [38]

Tabelle 5.6: Literaturergebnisse für die Evaluation

5.5 Workflow für erklärbare Systeme

Zusammenfassend ist auf Basis der Literaturergebnisse folgender Workflow entstanden (siehe Abbildung 5.6). In der Praxis liegt es im Ermessen der ausführenden Personen, wann sie welche Aktivität ausführen möchten. Eine Aktivität zählt verschiedene Methoden auf, die angewendet werden können. Hierbei müssen nicht alle Methoden angewendet werden, um das übergeordnete Ziel zu erreichen. Die Auswahl der Methodik basiert dabei auf dem zugrundeliegenden Kontext.
Eine Webseite würde sicherlich von einem Papier Prototyp profitieren, da es dadurch ersichtlicher wird in welchem Unterfenster welche Erklärbarkeitsproblematiken entstehen können. Für die Entwicklung eines erklärbaren Roboters ist ein Papier Prototyp gegebenenfalls eher nicht geeignet, da der Endnutzer hierbei keine Unterseiten bedient. Hier wäre ein Prototyp nützlicher, um eine Endnutzerevaluation durchführen zu können. Bei der Stakeholderanalyse kommt es bei der Wahl der Methoden darauf an, inwieweit Kontakte zu Endnutzern aufgebaut werden können. Ist es zu aufwendig Endnutzer zu rekrutieren, dann können fiktive Personas und Szenarios entwickelt werden, wobei darauf geachtet werden muss keine Biases in die Personas aufzunehmen und die Szenarios realitätsgetreu zu schreiben. Bestenfalls kann ein Fachexperte die Personas und Szenarios begutachten. Ansonsten eignen sich Interviews, Umfragen und Endnutzerobservierungen. Die Ergebnisse hieraus können wieder in Personas und Szenarios festgehalten werden, wobei sich die Gefahr der Biases und zu simplen bzw. nicht realitätsgetreuen Szenarios durch die Endnutzerinformationen reduzieren

kann. Weiterhin müssen auch nicht alle Aktivitäten angewendet werden. Beispielsweise kann eine Back-End Analyse nicht erforderlich sein, wenn keine komplexen Algorithmen im Back-End verwendet werden. Hierbei steht dann die Erklärbarkeit der Bedienung über die Oberfläche bzw. der Schnittstelle zum Endnutzer im Vordergrund. Daher lohnt es sich verstärkt die Zeit in das UI/UX Design zu investieren.
Es wurde ein Workflow entworfen, der in unterschiedliche Prozessmodelle eingegliedert werden kann. Von besonderer Bedeutung sind die Anforderungsanalyse und die Evaluation, da für diese Aktivitäten in wissenschaftlichen Texten die meisten Methoden verwendet werden. Letztendlich kann aber zusammengefasst gesagt werden, dass sich die Methoden aus dem RE und der MMK, die sich auch aus Methoden der Psychologie und Kognitionswissenschaft bedient, gut eignen, um die Anforderungen für erklärbare Systeme zu Erfassen. Sie eignen sich, um die mentalen Modelle der Endnutzer zu evaluieren und die Effektivität einer Erklärung zu messen. Da die Erklärbarkeit aber auch einen positiven oder negativen Einfluss auf andere Anforderungen haben kann, wie der Benutzerfreundlichkeit, sollten auch diese Anforderungen nach bekannter Vorgehensweise evaluiert werden. Da sich der Workflow aber auf Basis des Forschungskontextes bewegt, müssen die Faktoren aus der Industrie beachtet werden. Im folgenden Kapitel wird der erklärbare Workflow anhand der Ergebnisse aus den Experteninterviews evaluiert und im Kontext der Softwareentwicklungsprozesse in Unternehmen betrachtet.

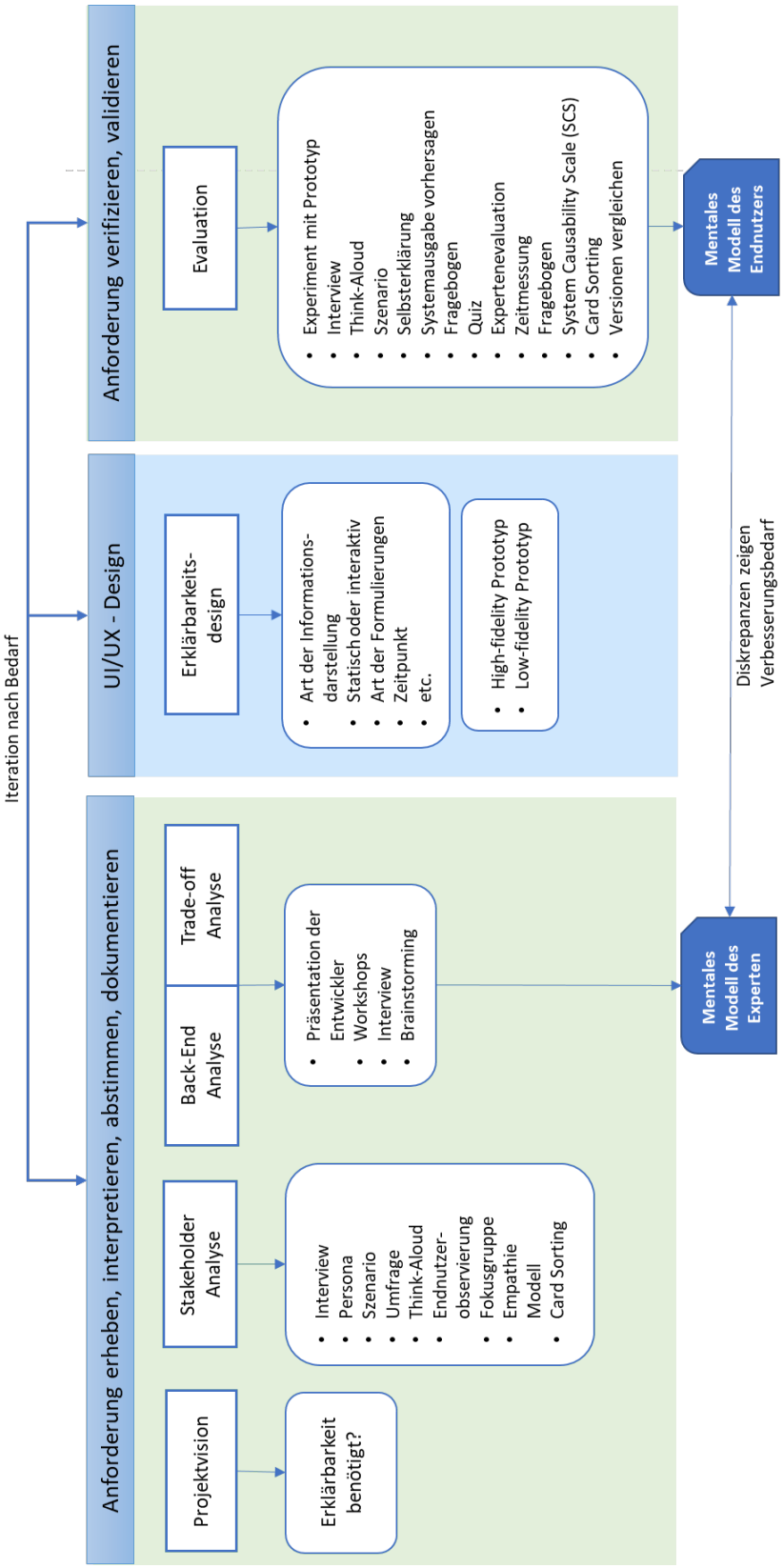

Abbildung 5.6: Workflow für erklärbare Systeme auf Basis der Literatur

Kapitel 6

Ergebnisse der Interviewstudie

Dieses Kapitel stellt die Ergebnisse der Experteninterviews dar. Dafür wurden die vorgeschlagenen Methoden pro Aktivität gezählt und in Balkendiagrammen in absteigender Sortierung dargestellt. Methoden, die nur durch einen Teilnehmer vorgeschlagen wurden, werden nicht aufgelistet, da keine Übereinstimmung mit anderen Experten vorliegt und die Eignung der Methode bezweifelt werden kann. Methoden die von weniger als vier Teilnehmern genannt wurden, werden mit einem hellblauen Balken dargestellt. Da nur eine Minderheit der Teilnehmer diese Methode empfehlen, wird in der textuellen Ausarbeitung die Methode nur beschrieben, wenn die Empfehlung ordentlich begründet wurde. Eine ordentliche Begründung geht über die geläufige Definition der Methode hinaus und stellt einen positiven Bezug zur Erklärbarkeit dar. Die Ergebnisse aus den Transkripten werden kategorisiert. Um die Beweggründe der Teilnehmer zu verdeutlichen, werden unterstützende Zitate verwendet. Abschließend werden weitere Erkenntnisse aus den Interviews dargestellt. Pro Aktivität wird die jeweilige Methode erläutert. Da eine Methode in unterschiedlichen Aktivitäten verschiedenen Anwendungskontexten unterliegt, können diese auch mehrmals erwähnt werden. Beispielsweise wird ein Interview in der Anforderungserhebung in einer anderen Art und Weise durchgeführt, als in der Evaluation. Dementsprechend existieren hierfür auch unterschiedliche Beweggrunde der Empfehlung. Zunächst werden die Ergebnisse der ersten Aufgabe aus dem Interview zusammenfassend dargestellt, in der die Teilnehmer im Unternehmen aktuell verwendete Methoden nannten.

Dieses Kapitel wird wie in der Aufgabenstellung für die Teilnehmer strukturiert, um eine Vergleichbarkeit der Ergebnisse zu erleichtern. Eine Besonderheit bezüglich der Aktivität *Anforderungsmanagement* ist jedoch, dass die Mehrheit der Teilnehmer die Anforderungsanalyse vom Anforderungsmanagement nicht abgrenzen konnte, weshalb der Interviewende diesen Schritt während der Aufgabenbearbeitung im Interview nocheinmal erklärte. Dennoch wurden keine Methoden genannt, die für das Anforderungsmanage-

ment geeignet wären, weshalb diese Aktivität nicht abgebildet wird. Ebenso konnten für die Aktivität *Kodierung* keine Methoden erhoben werden, was jedoch im Folgenden näher erläutert wird.

6.1 Aktueller Entwicklungsprozess

Alle Teilnehmer gaben an, in ihrem Unternehmen einen agilen Softwareentwicklungsprozess zu verwenden. Die kleinen Unternehmen verwenden dabei Scrum, die großen unterliegen dem V-Modell wobei SAFe[1] integriert wird. Dabei lobten sie die agile Angehensweise, da sie durch das iterative Vorgehen vieles unkomplizierter mache. Innerhalb einer Iteration wird zum Schluss getestet oder evaluiert. Betont wurde daher auch oft der Feedback-Loop, der essenziell sei:

> "*Fischer: ...[Ohne den] Feedback Loop...kann das Produkt nicht richtig entwickelt [werden]...Die Evaluation ist quasi die Feedbackstufe [mit der] eine neue Anforderung entwickelt wird...* "

Dokumentationen werden in kleinen Unternehmen nur selten vorgenommen, in großen Unternehmen sind sie aber verpflichtend. Um Anforderungen zu erheben, verwenden die Teilnehmer Interviews (84 %), Brainstorming mit Kunden und Kollegen (47 %), Szenarios (21 %), Workshops (21 %), Endnutzerobservierungen (21 %) und/oder Umfragen (21 %). Innerhalb der Anforderungsanalyse wurden Epics bzw. Use Cases (95 %) verwendet um die Anforderungen zu verfeinern und festzuhalten. In 79 % der Unternehmen werden Anforderungen anschließend priorisiert. Für den Entwurf werden Low-fidelity Prototypen (63 %) (wie Mock-Ups und Papier-Prototypen) und/oder High-fidelity Prototypen (21 %) entwickelt. In der Kodierung wurde nur die Dokumentation innerhalb des Programmcodes erwähnt (47 %). In der Evaluation werden Usability Tests (53 %), Endnutzerobservierungen (53 %) und/oder A/B Tests (53 %) angewendet.

Die Ergebnisse des aktuallen Entwicklungsprozess werden mit dem empfohlenen Workflow für erklärbare Systeme verglichen.

6.2 Entwicklungsprozess für erklärbare Systeme

Iterativer Entwicklungsprozess

Ein iterativer Entwicklungsprozess wurde von allen Teilnehmern als essenziell eingestuft, um erklärbare Systeme zu entwickeln. Dies liegt dadrin begründet, dass die Erfüllung einer Erklärbarkeitsanforderung stark subjektiv von den

[1] Das Scaled Agile Framework (SAFe) bindet agile Methoden in Organisationen ein, die aus mehreren Ebenen bestehen, wie beispielsweise multiple Entwicklerteams in unterschiedlichen Abteilungen innerhalb der Organisation oder des Unternehmens. [17]

Endnutzern und dem Kontext abhängt. Es existiert keine numerische Metrik die überprüft werden kann um zu versichern, dass das System erklärbar ist. Da die Erklärbarkeit noch recht unbekannt ist (nur zwei Teilnehmer haben das Wort *Erklärbarkeit* schoneinmal gehört), fehlt es an der nötigen Erfahrung wie erklärbare Systeme überhaupt aussehen sollen oder können. Dadurch muss ein Trial-and-Error Verfahren angewandt werden, um sich dem erklärbaren System schrittweise iterativ anzunähern. Anhand der Evaluationsaktivität kann mittels vorgestellter Methoden das Verständnis des Endnutzers überprüft und der ursprüngliche Designansatz überarbeitet werden, um die Erklärbarkeit des Systems zu verbessern. Ein Teilnehmer beschreib den Ablauf wie folgt:

> "*ID18: Ich würde das nach SAFe beziehungsweise nach Scrum machen,...du versuchst mit diesem Setting inkrementell vorzugehen...nicht wie beim Wasserfall...Du versuchst dich anzunähern an die perfekte Lösung.*"

Dieser Teilnehmer hebt hervor, dass der Wasserfallentwicklungsprozess aufgrund seiner sequentiellen Art eher nicht geeignet sei, um erklärbare Systeme zu entwickeln. Dies begründert er damit, dass eine perfekte Lösung in der Planungsphase eher nicht gefunden werden kann. Ein weiterer Teilnehmer sagte:

> "*ID05: ...man versucht erstmal herauszufinden, ob die These die man aufgestellt hat überhaupt im kleinen funktioniert.*"

Hier wird betont, dass die Pläne, bezüglich der Art und Weise wie die Erklärbarkeit umgesetzt werden sollen, zunächst nur eine These sind, die evaluiert werden müssen. Dies soll verhindern, Zeit in die Entwicklung eines Softwaresystems zu investieren, welches möglicherweise garnicht funktional ist. Ein weiterer Teilnehmer geht spezifischer auf Fragen ein, die innerhalb der Evaluierung aufkommen können. Auf diese sollen anhand einer Feedback-Loops in der Entwurfsaktivität entsprechend reagiert werden:

> "*ID01: ...reicht das, was du da erklärt hast, jetzt eigentlich schon?...versteht der Kunde das immer noch nicht?...Das musst du halt wieder verarbeiten, indem du dann deinen Entwurf anpasst...*"

Zusammenfassend sind die Teilnehmer einstimmig dafür, erklärbare Systeme anhand eines iterativen Prozesses zu entwickeln. Sie begründen dies mit der subjektivität der Erklärbarkeit, da nicht vorausgesetzt werden kann, dass ein Entwurf auch erklärbar für den Endnutzer ist. Da agile Vorgehensweisen iterativ sind und insbesondere dadurch Änderungen in einer umkomplizierten Art durchgeführt werden können, empfehlen ebenso alle Teilnehmer einen agilen Entwicklungsprozess.

Endnutzerzentrierter Entwicklungsprozess

Der endnutzertentrierte Entwicklungsprozess wurde von allen Teilnehmern für die Entwicklung erklärbarer Systeme empfohlen. Dadurch motiviert wurden für die einzelnen Aktivitäten hauptsächlich typische endnutzerzentrierte Methoden genannt wie Interviews, Personas, Umfragen und die Endnutzerobservierung. Im Folgenden wird genauer auf alle einzelnen Aktivitäten eingegangen und die Gründe beschrieben.

6.2.1 Anforderungsanalyse

In der Anforderungsanalyse wurden mehrheitlich endnutzerzentrierte Methoden genannt (siehe Abbildung 6.1) Mit diesen Methoden sollen die Bedürfnisse und der Kontext des Endnutzers analysiert werden. Ein Teilnehmer beschrieb die Anforderungsanalyse folgendermaßen:

> "*ID18: ...die Anforderungsanalyse [bzw. die] Anforderungen, [ist/sind] einfach eine Hypothese wie der Benutzer...in der User Story...mein System benutzen möchte.*"

Die erhobenen Anforderungen werden in der User Story als eine Hypothese definiert, wie der Endnutzer die Software verwendet. Diese muss im laufenden Prozess bestätigt oder widerlegt werden. Hierfür geben andere Teilnehmer an, dass es wichtig sei endnutzerzentrierte Methoden in der Anforderungsanalyse anzuwenden:

> "*ID04: ...in der Anforderungsanalyse [soll] auf jeden Fall eng mit Usern [zusammengearbeitet werden]...*"

> "*ID10: ...erklärbar ist was einigermaßen subjektives...Deswegen ist es wichtig, Methoden, die auch auf verschiedene subjektive Empfinden eingehen, anzuwenden.*"

Wie in der Abbildung 6.1 dargestellt, werden die empfohlenen Methoden für die Anforderungsanalyse im Folgenden vorgestellt.

Interview Das Interview ist mit 14 Nennungen die meist erwähnte Methode für die Anforderungsanalyse. Mit dem Interview können die Ziele, Wünsche, Bedürfnisse und Beweggründe der Stakeholder und Endnutzer analysiert werden. Antworten auf die Fragen *Für Wen*, *Warum* und *Was* erklärt werden soll, können gefunden werden. Weitehin können neue Erkenntnisse entdeckt und offene Fragen angesprochen werden. Zunächst sei es wichtig, den Endnutzer und seine Bedürfnisse kennenzulernen:

> "*ID04: ...[ein] offenes Interview angehen [und] da abfragt, was sie so brauchen, wollen. Wer der Endnutzer überhaupt ist.*"

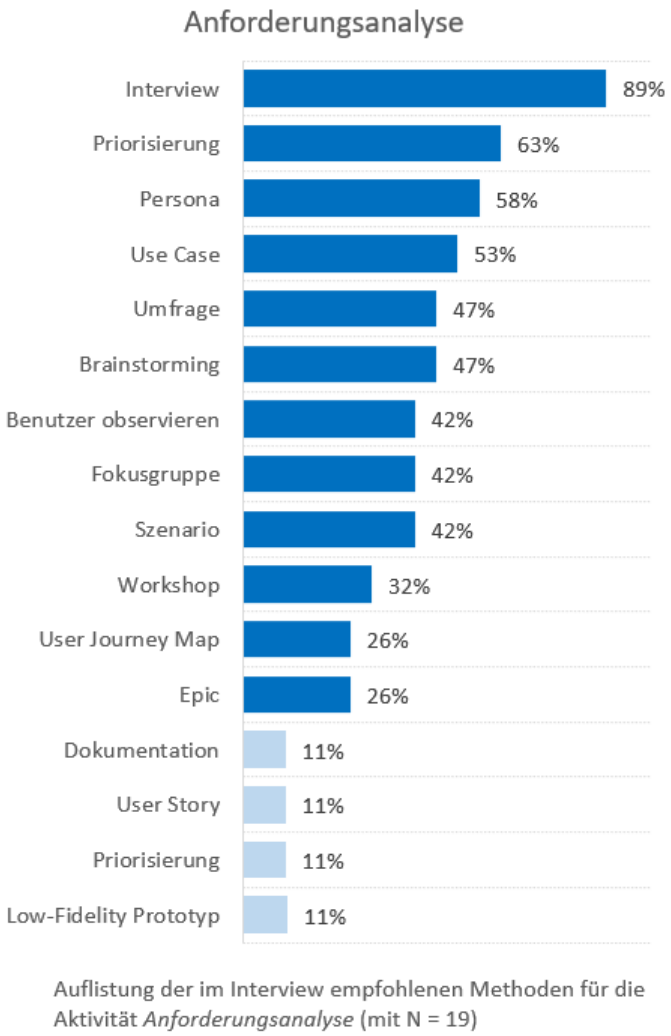

Abbildung 6.1: Interviewergebnisse für die Anforderungsanalyse

Weiterhin sagt ein Teilnehmer, dass analysiert werden muss, warum der Kunde erklärbare Software fordert und was er darunter versteht:

> "*ID01: ...erstmal [wollen wir] rausfinden, warum will der Kunde das eigentlich und was stellt er sich vor...einfach machst du das mit einem Interview...*"

Er fügt hinzu:

> "*...alle Leute erzählen...immer, dass sie das wollen...[aber] Wie wollen Sie das...?*"

Ein anderer Teilnehmer würde die Frage, *Was* erklärt werden soll durch direktes nachfragen innerhalb eines Interviews beantworten:

> "*ID06: ...[Im] Interview...fragen, was der Nutzer gerne hören möchte [was erklärt werden soll]...*"

Zusammenfassend soll mit einem Interview innerhalb der Anforderungsanalyse zunächst der Endnutzer oder Kunde kennengelernt werden, um ihre Beweggründe und Bedürfnisse festzustellen und ebenso ihre Vorstellung der Umsetzung nachvollziehen zu können.

Persona Die Persona wird von neun Teilnehmern empfohlen, da sie die Entwickler dabei unterstützt die Endnutzergruppen während der Entwicklung vor Augen zu halten und das Softwaresystem an sie anzupassen:

> "*ID04: Dann kann man [durch die Interviews] Personas entwickeln. Um überhaupt zu gucken, wem will ich da überhaupt die Erklärung geben?*"

Umfrage Die Umfrage wurde von acht Teilnehmern empfohlen. Umfragen wurden für die Anforderungsanalyse und Evaluation vorgeschlagen. Es kann sich hierbei um online oder offline Fragebögen handeln, die von Stakeholdern mit Informationen gefüllt werden. Sie ermöglichen ein standardisiertes Vorgehen und die quantifizierung der Ergebnisse. Zusätzlich können mit offenen Fragen explorative Ergebnisse erhoben werden. Mit präzise formulierten Fragen können die Ziele, Wünsche und Bedürfnisse der Stakeholder erhoben und analysiert werden.

> "*ID05: Umfragen [in denen gefragt wird] was...macht die Software erklärbar? Wann wundert man sich über Sachen, die [die] Software macht, um...rauszufinden, was sind denn die Szenarien und Use Cases, die wir...erklären [wollen]?*"

Endnutzerobservierung Die Endnutzerobservierung wurde von acht Teilnehmern empfohlen. Dabei soll der Endnutzer in einem typischen Anwendungskontext beobachtet werden. Der Beobachter kann analysieren in welchen Situationen eine erklärbare Software Verbesserung verspricht. Auch kann ermittelt werden, welche Systemausgaben Erklärungen benötigen und welche gegebenenfalls auch ohne auskommen können. Ein Teilnehmer sagte Folgendes dazu:

> "*ID03: Die Person den ganzen Tag [beobachten]. Wie kann man es [dem Endnutzer] leichter machen...Vielleicht sind auch bestimmte Ergebnisse nicht so wichtig wie andere.*"

Beispielsweise kann von einer Menge an Erklärungen eine Erklärung von besonderer Bedeutung sein, weshalb in der Entwicklungsphase das

Augenmerk auf diese Erklärung gelegt werden sollte. Gegebenenfalls benötigt der Endnutzer in bestimmten Kontexten auch keine Erklärung, da es in seinem Fachgebiet als grundlegendes Wissen eingeschätzt wird.

Brainstorming Das Brainstorming wurde von sieben Teilnehmern empfohlen. Mit dem Brainstorming können erhobene Anforderungen mit dem Kunden diskutiert werden, um Lösungsmöglichkeiten zu finden oder unterschiedliche Interpretierungen der Anforderung aufzudecken. Lösungsansätze zur Umsetzung können erörtert werden, um die Anforderungen zu verfestigen:

> "*ID06: [Um] rauszufinden, was möchte er (der Kunde) wirklich und ist er damit zufrieden?*"

> "*ID05: Brainstorming [halten] um Ideen zu kriegen...wie erklärt man das am besten? Was braucht man dafür?*"

Szenario Das Szenario wurde von sechs Teilnehmern empfohlen. Ein Teilnehmer ist auf die Problematik des nicht vorhandenen Akzeptanzkriteriums für die Erklärbarkeit eingegangen, mit der das Entwicklerteam die Erfüllung einer Anforderung normalerweise validiert. Seine Lösung hierfür ist ein Szenario:

> "*ID15: [Die Entwickler werden fragen] was ist das Akzeptanzkriterium, wann ist das für dich erklärbar?...dann würde ich...Szenarien definieren...und sagen: In einem solchen Fall wünsche ich mir..., dass ich eine Information über den Grund der geänderten Routenführung bekomme.*"

Szenarien werden auch als eine explorative Methodik angesehen, mit der die verschiedensten Geschehnisse festgehalten werden, um angepasste Lösungen zu entwickeln:

> "*ID05: Was gibt es eigentlich für unterschiedliche Szenarien?*"

> "*ID19: ...durchgehend verschiedene Szenarien entwickeln.*"

Use Case Use Cases sind ein Hauptbestandteil in der Softwareentwicklung. Sie beschreiben die Softwarefunktion in einer detaillierten Art und können für die Erklärbarkeit hilfreich sein, da sie das Vorgehen eines Systems und die Interaktionsmöglichkeiten schrittweise darstellen. So kann pro Funktion darüber entschieden werden, ob eine Erklärung notwendig sein kann und wie eine passende Erklärung aussehen könnte:

> "*ID14: ...wir haben ein Epic...das sagt das Fahrzeug soll bei [einem] Unfall ausweichen auf eine andere Route. Dann [hat man] ein Use Case, durch das...früh genug berücksichtigt...[wird]...ein Mockup zu machen, [um] das Ganze zu visualisieren...[oder] die Entscheidung [zu] fällen....teilen wir die Informationen dem User mit oder [nicht]?.*"

Fokusgruppe Fokusgruppen wurden von fünf Teilnehmern empfohlen. Die Vorteile bezüglich der Erklärbarkeit sind, dass mit den Endnutzern eine Diskussion geführt werden kann in der Lösungsansätze gemeinsam erörtert werden können:

> "*ID10: Da können wir dann gemeinsam in der Diskussion...die optimale Lösungen erörtern*"

Weiterhin können mehrere Fokusgruppen errichtet werden, in denen eine bestimmte Gruppe von Endnutzern ihre Bedürfnisse, Ziele und Bedenken mitteilt. Für die Effektivität einer Erklärung kann beispielsweise die Altersgruppe ein ausschlaggebender Faktor sein. Unter anderem unterscheidet sich der sprachliche Gebrauch von jungen und älteren Personen, weshalb die Formulierungsart der Erklärung eine Gruppe bevorzugen könnte. Auch durch das erwartete Vorwissen über bestimmte Bereiche kann eine Erklärung alters- oder berufsabhängig gestalten:

> "*ID06: Fokusgruppe...ist...interessant im Hinblick auf verschiedene Nutzergruppen. Bei jüngeren Leuten, zum Beispiel von 20 bis 30...dann muss man...testen, welche Erklärungen braucht welche Altersgruppe?...Vielleicht hängt es mit dem Beruf zusammen. Manche Leute wollen ein bisschen einfacher die Sachen erklärt haben als Ärzte oder Anwälte.*"

Insbesondere in diesem Fall können unvorhergesehene Anforderungen aufgedeckt werden.

Trade-of Analyse Eine weitere Besonderheit stellte der Fakt dar, dass die Trade-of Analyse nur von zwei Teilnehmern selbstständig genannt wurde. Die anderen Teilnehmer wurden am Ende der zweiten Aufgabe befragt, ob es ihrer Meinung nach eine negative Abhängigkeit der Erklärbarkeit mit anderen Anforderungen geben könnte. Die Teilnehmer antworteten, dass dies der Fall sei. Ein Teilnehmer erklärte, dass unter Umständen sogar weitreichende Konsequenzen auftreten können:

> "*ID19: Beispielsweise wir mit unseren Stakeholdern sagen okay, wir müssen...drei Sekunden bevor das Auto abbiegt eine Erklärung anzeigen. Das finden wir irgendwie sinnvoll, aber in der*

> *Evaluation kommt raus, dass der Nutzer gerne zehn Sekunden vorher Bescheid kriegt, damit er auch Zeit hat das zu lesen [und] zu verstehen. Das ist eine Anforderung die eventuell weitreichende Konsequenzen hat. Weil vielleicht die Sensordaten nicht schnell genug sind...[um die] Antwortzeit...von drei Sekunden auf zehn Sekunden [zu] verlängern, damit ich meinen Endnutzer zufrieden mache...*"

Es stellte sich herraus, dass die Teilnehmer die Trade-off Analyse nicht erwähnt haben, da dieses Problem mit jeder Anforderung bestehe. Da es kein explizites Problem der Erklärbarkeit darstelle, sondern ein allgemeines (beispielsweise Security und Usability können auch in negativer Abhängigkeit zueinander stehen) haben sie diese Methode nicht erwähnt, obwohl sie zutrifft:

> "*ID18: Das ist ja in den meisten Fällen so, dass du nicht-funktionale Anforderungen...in gewissen Bereichen nicht durchführen kannst, weil du gewisse Constraints hast, um das umzusetzen. Oder...auf Basis deiner funktionalen Anforderungen...gewisse nicht-funktionale Anforderungen nicht durchführen [kannst]...Die beste User Experience ist, wenn du...das per Audio erklärst, weil er das besser aufnehmen kann [anstatt eines]...Textes...du hast aber in deinem System kein Audio zur Verfügung, dann kannst du die...nicht-funktionale Anforderung nicht umsetzen.*"

Der Teilnehmer erwähnte den Trade-off bezüglich der Benutzererfahrung durch die Erklärbarkeit in Bezug auf die Systemhardware. Ein Text der gelesen werden muss, senke dementsprechend die Benutzererfahrung. Gegebenenfalls könnte es in einigen Fällen daher bedeuten, dass für die optimale Umsetzung der Erklärbarkeit, wenn diese denn höchste Priorität hat, auch Anforderungen bezüglich der Hardware des Systems beeinflusst werden.

6.2.2 Softwaredesign

In der Aktivität Entwurf haben die Teilnehmer hauptsächlich zu einem Low-fidelity Prototypen geraten (siehe Abbildung 6.2). Nachdem in der Anforderungsanalyse ermittelt wurde, *Was* erklärt werden soll, wird in der Entwurfsaktivität die Art und Weise definiert, *Wie* die Erklärungen dem Endnutzer zu verfügung gestellt werden sollen. Ein Teilnehmer findet es wichtig, dass nicht zu viel erklärt wird, da es die Wahrnehmung der Endnutzer für die wichtigen Erklärungen vermindere:

> "*ID13: ...Ich finde es wichtig, dass [in den] Entwurf nicht zu viele Informationen kommen, weil [der Endnutzer diese] dann*

auch irgendwann nicht mehr wahrnehmen wird, [da] es einfach zu viel ist...er [sollte] nur die notwendigen oder für ihn tatsächlich interessanten Informationen [erhalten.]"

Ein weiterer Teilnehmer sagt, dass der Endnutzer auch den Grund der Erklärung verstehen sollte:

"*ID06: Am besten ist [es], wenn er auch genau weiß, warum er dann diese Info bekommt, also [er sollte] nicht nur [die Erklärung verstehen], sondern auch warum.*"

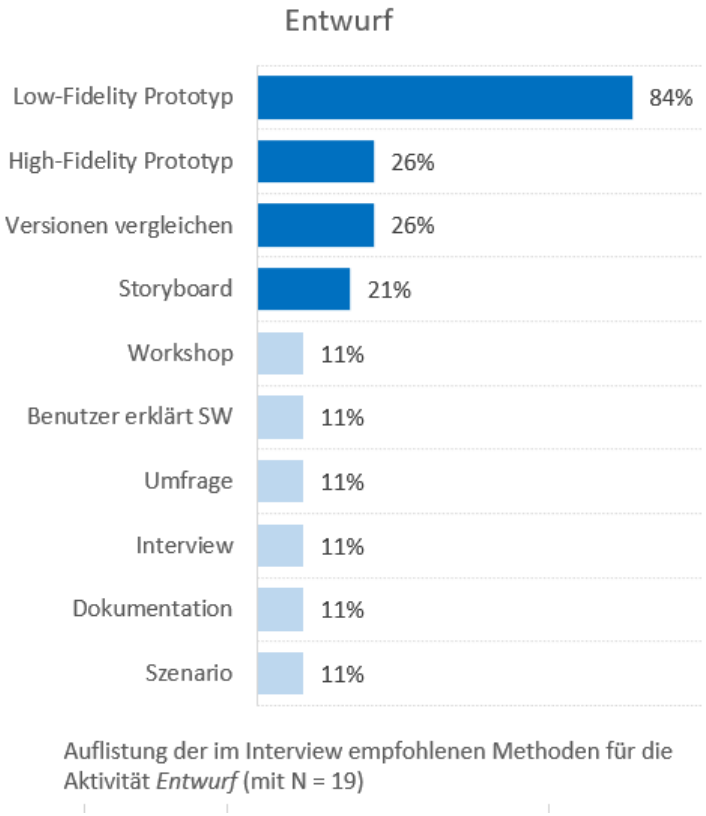

Abbildung 6.2: Interviewergebnisse für den Entwurf

Low-fidelity Prototyp Low-fidelity Prototypen wurden von 16 Teilnehmern empfohlen. Davon würden 15 Teilnehmer Mock-Ups und sechs Wireframes verwenden. Da erklärbare Systeme iterativ entwickelt werden sollen, eignen sich Low-fidelity Prototypen gut um schnell Designentscheidungen zu evaluieren und mehrere Versionen zu erstellen die verglichen werden können.

> "*ID19:...Prototyp oder Mock-up je nachdem was ich gerade genau möchte, also es kommt...immer völlig auf das Feature an. [Oder] auch Papier-Prototypen. Also...ich entwerf erstmal was [um zu] gucken, was da rauskommt. Im Sinne von dem Auto [macht] Wireframe...nicht unbedingt Sinn.*"

Es wurde als ein schnelles, günstiges und interaktives Verfahren betitelt die sich dennoch gut dafür eignet den geplanten Softwareentwurf mit Stakeholdern oder die Entwickler unter sich zu testen. Hier fallen die Problematiken bezüglich der Erklärbarkeit auf und können schnell mit einer Anpassung des Mock-Ups behoben werden und neu getestet werden.

High-Fidelity Prototyp High-fidelity Prototypen werden von fünf Teilnehmern empfohlen. Obwohl in der Literatur Prototypen oft verwendet werden, verwenden die Teilnehmer aus den kleinen Unternehmen kaum High-fidelity Prototypen. Begründet wird dies, da es zu aufwendig und zeitintensiv sei einen funktionalen Prototyp zu erstellen.

> "*ID01: ...Einen klassischen Prototypen,...den gibt es bei uns nicht. Wir überlegen uns das nachher so weit, dass wir das tatsächlich auf live schieben und dann gucken was kriegen wir für Feedback...*"

Hierbei kommt es aber darauf an, welche Art von Softwaresystem erstellt wird. Webseiten können sicherlich direkt veröffentlicht werden, aber im Kontext des autonomen Autos aus dem Fallbeispiel muss dieser mit einem High-fidelity Prototypen zuerst getestet werden, bevor dieser der Allgemeinheit zugänglich gemacht wird. Es entstand der Eindruck, dass die Teilnehmer keinen direkten Bezug zu High-fidelity Prototypen haben, da dieser innerhalb großer Unternehmen in anderen Abteilungen erstellt und getestet wird, während die Teinehmer der Interviewstudie aus kleineren Unternehmen direkt die Softwareanwendung veröffentlichen, um die Änderungen zu testen.

Storyboard Ein Storyboard würden vier Teilnehmer im Softwaredesign erstellen. Anhand eines Storyboards kann der Durchlauf des Endnutzers in der folgenden Evaluationsaktivität durchleuchtet werden:

> "*ID01: Wie geht der Kunde da eigentlich durch?*"

Dadurch können spezielle Aspekte des Softwaresystems auf die Reaktion des Endnutzers geknüpft werden. Wenn ein Endnutzer während der Verwendung des Softwaresystems sich beispielsweise negativ äußert, dann können schlechte Designentscheidungen schneller erfasst werden. Aber auch für die Entwickler sind Storyboards hilfreich. Sie können sich an ihnen orientieren, da die Funktionsweise des Softwaresystems nachvollziehbar dargestellt wird:

> "*ID06: [Damit man] eben einfach diesen ganzen Prozess im Überblick hat, dafür ist ein Storyboard gut.*"

Gegebenenfalls könnten Storyboards auch als eine Art Kommunikationsmittel für ein einheitliches Verständnis innerhalb des Entwicklerteams verwendet werden.

6.2.3 Kodierung

Für die Kodierungsphase wurde mehrheitlich keine Methode genannt die explizit einen Mehrwert für erklärbare Systeme erbringen würde.

> "*ID19: Also Kodierung ist letztlich nur die Umsetzung*"

Die Dokumentation wurde von fünf Teilnehmern genannt. Hiermit wurde die Dokumentation des Programmcodes gemeint, mit der die Entwickler die Funktionsweise des Back-Ends mit den Front-End Entwicklern kommunizieren können. Lediglich eine Methode wurde selbstständig von zwei Teilnehmern empfohlen, das *Pair Programming*. Diese Methode eignet sich laut der Teilnehmer, innerhalb agiler Prozesse da sich Front-End und Back-End Entwickler ihr jeweiliges Wissen über die Funktionsweise des Back-Ends weitergeben können. Dies ist von besonderer Bedeutung, da Front-End Entwickler letztendlich diejenigen sind, welche die Erklärungen entwerfen und in das System einbinden:

> "*ID15: Ich befürworte sicherlich auch, dass im Entwicklerteam Leute auch mal Dinge tun auf Kosten der Effizienz und der Effektivität, die nicht ihr Steckenpferd sind ... lernen ... ist dann halt für mich der Mehrwert.*"

Weiterhin sagte ein anderer Teilnehmer:

> "*ID18: Wenn man zu zweit drauf schaut hab ich dann direkt Feedback...tauschen wir uns dann aus und krieg verschiedene Perspektiven mit, als nur vom Code. Also ich krieg ein allgemeines Verständnis der Software und wie sie zu funktionieren hat.*"

Weiterhin wurden die Teilnehmer befragt, inwiefern die Kommunikation der Front-End Entwickler mit den Back-End Entwicklern bezüglich der algorithmischen Funktionsweise des Systems ihrer Meinung nach wichtig ist. Diese Frage zielt auf die Evaluierung der Aktivität *Back-End Analyse* ab, in der die Software auf algorithmischer Basis den Rollen erklärt wird, die sich mit der Integrierung der Erklärbarkeit für den Endnutzer beschäftigen, wie unter anderem Front-End Entwickler und Designer. Die Mehrheit beantwortete diese Frage mit „Ja“. Nur drei Teilnehmer fanden die Kommunikation beider Rollen für die Entwicklung erklärbarer Systeme nicht

wichtig. Die Frage wurde gestellt um zu erfassen ob Front-End Entwickler ein weitergehendes Verständnis für erklärbare Systeme benötigen, da sie diese potenziell zum Endnutzer anhand der Erklärbarkeit weiterkommunizieren müssen.

> "*...um das überhaupt darstellen zu können, muss der Back-End Entwickler ihm erstmal diese Information geben...Und deshalb müssen die das zusammen machen, weil sie sich einfach austauschen müssen.*"

Dafür wurden innerhalb kleiner Unternehmen Gespräche vorgeschlagen. Für große Unternehmen sollen sich online Kommunikationsmethoden durch eine mögliche räumliche Trennung eher eignen:

> "*ID10: ...vielleicht [sitzen sie]sogar in zwei verschiedene Firmen...aber ich würde...kommunizieren...über Tickets [oder] Systemchannels in MS Teams oder Slack App zum Beispiel.*"

Ein anderer Teilnehmer erwähnte die Möglichkeit erklärende Videos zu erstellen um mit anderen Abteilungen zu die Funktionsweise der Software zu kommunizieren:

> "*ID13: regelmäßig...unter Gruppen vorstellen und...kleine Videos machen, was gemacht wird*"

6.2.4 Evaluation

Innerhalb der Evaluation wurden die meisten Methoden empfohlen (siehe Abbildung 6.3), da die Effektivität der Erklärbarkeit nicht innerhalb von Systemtests gemessen werden kann. Aufgrund der Subjektivität ist das Entwicklerteam auf die Rückmeldung der Endnutzergruppen angewiesen:

> "*ID19: ...[man braucht] Feedback, weil Erklärbarkeit...etwas subjektives [ist]. Verschiedene Personengruppen würden andere Antworten geben, ob ein System erklärbar ist oder nicht.*"

Dies kann durch die endnutzerzentrierten Methoden erreicht werden:

> "*ID10: [Durch die] Benutzerobservierung, Tester [und] Usability Tests bekommen [wir] sehr gute, sehr subjektive Rückmeldungen von jedem einzelnen Nutzer...[Daraus] kann man gut ableiten...haben wirklich viele Benutzer Probleme gehabt das hier zu verstehen oder war das vielleicht nur bei den Einzelnen ein Problem?..*"

Aber nicht nur subjektive Rückmeldungen sind wichtig, auch quantitative Ergebnisse sind wichtig für die Evaluation von erklärbaren Systemen:

"*ID04: [ich würde] die Performanz mit messen, weil es nicht nur wichtig ist, ob dem User das gefällt, sondern auch ob er sich selber besser verhält...*"

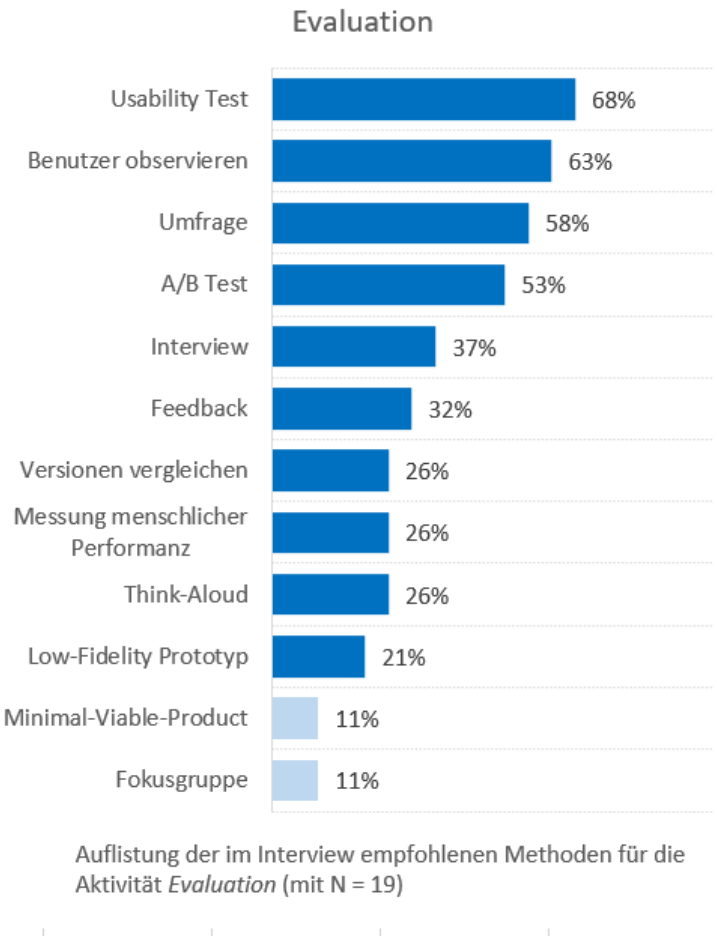

Abbildung 6.3: Interviewergebnisse für die Evaluation

Usability Test Der Usability Test wurde von 13 Teilnehmern empfohlen. Einige Teilnehmer sagten, dass die Erklärbarkeit einen Faktor innerhalb der Usability darstelle und daher ein Usability Test sich gut für die bestimmung der Qualität der Erklärbarkeit eignet. Weiterhin kann mit einem Usability Test überprüft werden, ob durch die Integrierung der Erklärbarkeit ein Trade-off mit der Usability entstanden ist. Ein Usability Test mit eher negativen Ergebnissen kann darauf hinweisen, dass die Erklärbarkeit nutzerfreundlicher gestaltet werden muss. Die Verwendung eines Usability Tests setzt einen Prototyp oder eine Simulation vorraus.

Endnutzerobservierung Die Endnutzerobservierung wurde von zwölf Teilnehmern empfohlen. Begründet wurde dies mit der Möglichkeit den Benutzer während der Verwendung des Softwaresystems zu beobachten um Probleme bei der Verwendung und des Verständnisses zu überblicken. Dadurch können auftretende negative Emotionen des Endnutzers mit der Softwarefunktion verknüpft werden, die diese erzeugt hat. Der problematische Schritt oder die Funktion kann so erfasst und überarbeitet werden. Ein Teilnehmer zieht Bezug auf die Hypothese die innerhalb der Anforderungsanalyse aufgestellt wurde, um diese zu belegen bzw. zu widerlegen:

> "*ID18: [Dabei] schaut [man] wie reagiert er jetzt gerade auf diese Situation? Das würde ich dann gegenhalten mit meiner Hypothese, die ich aufgestellt habe in meiner Anforderung.*"

Umfrage Die Umfrage wurde von elf Teilnehmern empfohlen. Mit ihr können die Endnutzer befragt werden, inwiefern sie mit der Erklärbarkeit des Softwaresystems zufrieden sind. Ein Teilnehmer spricht den unterschiedlichen Blickwinkel an, die das Entwicklerteam auf die Software hat:

> "*ID14: ....Das ist das Problem in der Softwareentwicklung. Dass man sich so stark mit der Software beschäftigt, dass man diese Aspekte vergisst...Hat der Kunde es wirklich verstanden? Deswegen [sollte] man...eine Umfrage...auf jeden Fall hier [machen].*"

Dabei reicht es nicht aus direkt nach dem Verständnis zu fragen. Ein Teilnehmer sagte, dass der Endnutzer erklären soll, wie er die Software verstanden hat, um diese auf Korrektheit zu bewerten:

> "*ID04: [Es können] Fragen wie "Weißt du, warum das Fahrzeug da und nicht da lang gefahren ist? "[gestellt werden.]*"

Aber nicht nur das Verständnis, sondern auch Daten zur Benutzererfahrung können mit Umfragen erhoben werden:

> "*ID04: In der Evaluation [können] Umfragen [gehalten werden], wie es [den Endnutzern] gefallen hat. [Damit man] messen kann, wie ist das Vertrauen [der Endnutzer der Software gegenüber]?*"

A/B Test Der A/B Test wurde von zehn Teilnehmern empfohlen. Der Vorzug für die Erklärbarkeit ist die Vergleichbarkeit von unterschiedlichen Designansätzen durch zwei zueinander ähnlichen Endnutzergruppen:

> "*ID04: Ich finde persönlich immer A/B Tests sehr sehr interessant, weil ich dadurch relativ einfach ähnliche Nutzergruppen vergleichen kann.*"

> "*ID05: A/B Tests sind sinnvoll, also zumindest, wenn man unterschiedliche Ideen hat wie man das Ganze angehen will.*"

Da die Teilnehmer in Bezug auf die A/B Tests oft von der Mehrzahl gesprochen haben (beispielsweise *Ideen* und *Änderungen*), hat der Interviewende vorsichtshalber nachgefragt, wie die Teilnehmer A/B Tests definieren. Dabei wurde ausgesagt, dass es sich um zwei Versionen handeln kann, die mehr als nur eine Aspektänderung beinhalten. Dies ist widersprüchlich gegenüber der Definition in der Literatur. In der Literatur soll mit einem A/B Test nur ein bestimmter Aspekt geändert werden, damit eine vorher aufgestellte Hypothese, die genau auf einen bestimmten Aspekt der Software bezogen ist, belegt oder widerlegt werden kann. [63] Daher ist fraglich, ob mit A/B Tests möglicherweise auch die Methode *Versionen vergleichen* beinhaltend gemeint wurde. Trotz der hierauf bezogenen Unwissenheit, kann jedoch daraus die Erkenntnis gezogen werden, dass Versionsvergleiche für die Entwicklung von erklärbaren Systemen äußerst nützlich sind.

Interview Das Interview wurde von sieben Teilnehmern für die Evaluation empfohlen. Ähnlich wie mit einer Umfrage kann die Meinung und die Erfahrung des Endnutzers nach der Verwendung des erklärbaren Systems erfragt werden. Durch das persönliche Gespräch kann der Grund für eine Antwort detaillierter hinterfragt werden. Insbesondere die Meinung und Erfahrung des Endnutzers während der Verwendung des Systems wurde von den Teilnehmern angesprochen:

> "*ID18: Wie hast du das Ganze gesehen?...Wie empfindest du das Ganze und hilft es dir? Also da hat man ja gewisse Kriterien, die man dann abfragt...*"

Auch ein weiterer Teilnehmer formulierte beispielhafte Fragen, die in einem Interview gestellt werden können, um die Erklärbarkeit des Systems zu beurteilen:

> "*ID19: Was meint du, aus welchem Grund hat das Auto so und so entschieden? Können Sie mir erklären [warum?]... Kannst du [dies] gut nachvollziehen? Ist das gut dargestellt? Oder: Kann man an der Erklärung was optimieren? Ist...der Hinweis für den Unfall auf dem Display groß genug dargestellt oder müsst ihr früher gewarnt werden?*"

Außerdem sagten mehrere Teilnehmer, dass der Interviewer gewisse Kompetenzen aufweisen sollte, um genaue Ergebnisse zu erzielen:

> "*ID19: Wirklich einen der geschult ist, der im Endeffekt weiß...wie man ein gutes Interview führt.*"

Versionen vergleichen Fünf Teilnehmer würden unterschiedliche Prototypversionen erstellen, um Designentscheidungen zu vergleichen. Beispielsweise könnten mehrere Ideen im Entwurfsprozess entstehen, die durch eine Endnutzerevaluierung bewertet werden. Da es zu Beginn mehrere Designansätze geben wird, wie die Erklärungen in das Softwaresystem integriert werden sollen, empfiehlt ein Teilnehmer die verschiedenen Versionen mit Endnutzern zu evaluieren, sich für eine zu entscheiden und diese weiter zu verbessern:

> "*ID06: Wenn man verschiedene Erklärungen aufgebaut hat, dann kann man testen, welche kommen am besten beim Nutzer an? [Um] sich dann am Ende für eins [zu] entscheiden.*"

Think-Aloud Das Think-Aloud wurde von fünf Teilnehmern empfohlen, wobei das Think-Aloud oftmals mit der Benutzerobservierung verknüpft wurde.

Low-fidelity Prototyp Das Low-Fidelity Prototyp wurde von vier Teilnehmern empfohlen. Hierbei war die Verwendung eines Low-Fidelity Prototypen, wie beispielweise ein Mock-Up, bei der Evaluation gemeint und nicht die Erstellung. Erst durch Verwendung eines Prototyps können die vorher genannten Methoden überhaupt angewendet werden. Während in den ersten Iterationen ein Low-Fidelity Prototyp vorteilhaft ist, da Änderungen unkomplizierter vorgenommen werden können, ist in fortgeschrittenen Iterationen ein High-Fidelity Prototyp anwendbar, welcher die letzliche Nutzererfahrung auch genauer simulieren kann.

6.2.5 Weitere Punkte

Neben den Methoden sind auch weitere Aspekte genannt worden die für die Entwicklung von erklärbaren Systemen für die jeweligen Teilnehmer wichtig sind.

Das Mindset

Drei Teilnehmern sagten, dass zunächst das richtige Mindset für die Erklärbarkeit bestehen muss.

> "*ID15:...Als Product Owner [muss ich] die Erklärbarkeit als ein Mehrwert definieren und meinem Entwicklerteam vermitteln...*"

Der beste Entwicklungsprozess für erklärbare Systeme würde nicht zum Ziel führen, wenn dem Team die Wichtigkeit der Erklärbarkeit nicht bewusst sei. Daher sollte der Teamleitder oder Product Owner die Motivation für die Integrierung der Erklärbarkeit dem Team erläutern und sie bestenfalls

dafür begeistern. Genauso wie die Anforderung der Sicherheit ständig berücksichtigt wird, muss auch die Erklärbarkeit bei jeder Änderung oder Anpassung im Hinterkopf behalten werden.

> "*ID04: Erklärbarkeit...so als Wort...können sich Leute relativ wenig drunter vorstellen...Man muss aber auf jeden Fall das auf dem Schirm behalten [und] sich mit nicht-funktionalen Anforderungen beschäftigen...Sobald man...sich [darum] konkret kümmern will, dann denke ich, ist der Prozess auf jeden Fall auch geeignet dafür.*"

Einfluss des aktuellen Prozesses

Auffalend war, dass alle Teilnehmer ihren aktuellen Softwareentwicklungsprozess des Unternehmens als Grundlage für den erklärbaren Workflow verwendet haben. Ein anderer Teilnehmer sagte, dass die Erklärbarkeit nur eine Anforderung sei, die es zu erfüllen gelte und der aktuelle Unternehmensprozess sich gut dafür eigne:

> "*ID15: Genau so! [Wie der aktuelle Unternehmensprozess] Die Erklärbarkeit ist nur eine Anforderung, die es gilt im Entwicklungsprozess zu erfüllen.*"

Die Teilnehmer haben zu Beginn der Bearbeitung der Aufgabe erwähnt, dass ihr aktueller Entwicklungsprozess sich gut eignen würde, um erklärbare Systeme zu entwickeln. Dies könnte darin begründet sein, dass der Prozess ihnen vertraut ist und der erste Gedankengang die Anpassung des eigenen Prozesses für erklärbare Systeme war. Ein anderer Grund kann sein, dass die Entwicklung von erklärbarer Software nicht vom Softwareentwicklungsprozess abhängt, solange die wichtigen Merkmale der iterativen und endnutzerzentrierten Entwicklung erfüllt werden:

> "*ID11: From what I put above in the 1st frame I wouldn't change much to be honest. I would try to just force it more towards the end user as much as possible... how we're doing things it's not me being very sure of my company doing the best thing, but I think moving more towards the more agile and moving more towards the end user would be the best way to do it because we have a certain perception ourselves how it should be explained with how it should be done, but it covers only ours.*"

Dies würde beispielsweise den Wasserfallentwicklungsprozess als nachteilig für erklärbare Systeme einstufen, da hier kein iterativer Entwicklungsprozess möglich ist und Anpassungen des Systems als problematisch angesehen werden.

Ein Teilnehmer erwähnte, dass die Entwicklung erklärbarer Systeme unabhängig von dem Softwareentwicklungsprozess sei und diese auch innerhalb eines Wasserfallentwicklungsprozesses möglich ist, aber eine längere Entwicklungszeit beanspruche im Vergleich zu agilen Vorgehensweisen:

> "*ID18: Ich hab ein bisschen das Problem zu verstehen, wie der Prozess da entscheidend ist, weil letztlich kann ich erklärbare Software...nach dem Wasserfallmodell bauen oder nach Scrum bauen...Beim Wasserfall dauert es halt länger, um gute Ergebnisse zu erreichen, glaube ich. Deswegen eignet sich halt so ein agiler Prozess hier deutlich besser...*"

Da alle Teilnehmer sagten, dass sie nach ihrem aktuellen Entwicklungsprozess erklärbare Systeme entwickeln würden, wurde ihnen die Frage gestellt, ob sie optimierungsbedarf in ihrem derzeitigen Entwicklungsprozess sehen, um erklärbare Systeme zu entwickeln. Alle Teilnehmer antworteten, mehr endnutzerzentrierte Methodiken zu integrieren um die Endnutzeranforderungen erfassen und Designentscheidungen auf die Endnutzerbedürfnisse hin evaluieren zu können.

Dazu sei erwähnt, dass alle Teilnehmer in ihrem Unternehmen einen agilen Entwicklungsprozess verfolgen. Dadurch lässt sich annehmen, dass agile Entwicklungsprozesse im allgemeinen gut geeignet sind um erklärbare Systeme zu entwickeln. Die Teilnehmer fügten endnutzerzentrierte Methodiken ihrem aktuellen Unternehmensprozess hinzu, wodurch ersichtlich wird, dass ein endnutzerzentrierter, agiler Entwicklungsprozess für die Entwicklung von Qualitativen erklärbaren Systemen, basierend auf den Ansichten der Experten, gut geeignet ist. Ein Teilnehmer hebte die Einbeziehung der endnutzerzentrierten Vorgehensweise durch agile Vorgehensweisen hervor:

> "*ID11: since I started working with the agile methodology, which was four years ago I've noticed that if it's done correctly, if the customer is involved, if a lot of UX is implemented...That there is not much that needs to be changed...As closer as you can get to the end user the better information you'll have and the better the end result will be because they will be directly involved with the process.*"

Ein weiterer Teilnehmer hebte die inkrementelle Entwicklungsart und die gute Anpassung an den Markt hervor:

> "*ID18: das schöne Vorgehen von agilen Methoden ist, dass du inkrementell vorgehst. Du entwickelst, passt das an und kannst es direkt den Kunden in die Hand geben als fertiges Produkt. Du [kannst] den Kunden beobachten und Step by Step dein System anpassen. Du bist viel schneller auf dem Markt, hast eine bessere*

> *Skalierung, viel bessere Anpassung [an den Markt]. In diesen Prozessschritten ist das Scrum Team sehr oft nicht involviert sondern, und das ist das...Problem...weil in diesen Schritten Wissen verloren geht.*"

Er fügt aber hinzu, dass es auch negative Seiten gibt:

> "*ID18: ...In den Prozessschritten [hier gemeint: Feedback] ist das Scrum Team sehr oft nicht involviert,[das ist das]...Problem...weil [dadurch] Wissen verloren geht.*"

Dadurch, dass das Scrum Team in der Endnutzerevaluierung nicht involviert ist, können sie nicht vollkommen nachvollziehen, weshalb Probleme in der Endanwendung entstanden sind. Mehrere Teilnehmer wünschen sich daher, dass sie an der Evaluationsphase teilhaben können, um ein tiefergehendes Verständnis zu entwickeln und ebenso schneller Feedback erhalten.

Realitätstauglichkeit

Abschließend wurden die Teilnehmer auf die Realitätstauglichkeit des von ihnen entworfenen Workflows für erklärbare Systeme befragt. Beispielhaft wurden Ressourcen wie Zeitaufwand, Kostenaufwand oder die Anzahl der Beschäftigten in einem Unternehmen als mögliche Einflussfaktoren auf die Realitätstauglichkeit vom Interviewenden genannt. Alle Teilnehmer antworteten, dass der Workflow auch in Bezug auf diese einschränkenden Einflussfaktoren realitätstauglich sei. Drei Teilnehmer merkten an, dass einpaar Methoden gegebenenfalls entfernt werden können, wenn das Unternehmen begrenzte Ressourcen hat. Im Endeffekt sollen aber endnutzerzentrierte Methoden unbedingt ausgeführt werden, damit die Erklärbarkeit des Systems für den Endnutzer gewähleistet werden kann:

> "*ID19: Dieses ganze A/B Testing, Nutzerfeedback [kann] auch weglassen [werden]. Die Frage ist: Hat man am Ende ein erklärbares System oder für wen ist das ausreichend erklärbar?...Der Auftraggeber würde sagen: Ja ist doch erklärbar. Aber ob das die Endnutzer dann für erklärbar halten das ist halt...ne?*"

Einige Teilnehmer erwähnten auch, dass der Workflow zunächst in der Anwendung getestet werden muss, um Erfahrungen aufzubauen und ihn gegebenenfalls anzupassen:

> "*ID10: Also ich find schon, dass das so durchführbar ist, weil es ist vor allem auch Wert zum Schluss. [Es macht] vonm Kostennutzenverhältnis schon Sinn. Ich würde am Anfang so alles verwenden...bei weiteren Iterationen [kann man] auswählen, was hat denn am besten geholfen oder wo hat es gehakt?...*"

> "*ID13: Ja, man muss halt Erfahrung sammeln und gucken was wie geeignet ist...aber ich denke schon, dass es auch notwendig ist.*"

Ein Teilnehmer erwähnte auch, dass die Größe des Unternehmens ein ausschlaggebender Faktor für die Umsetzung des Workflows darstellt:

> "*ID01: ...Ich hab den jetzt so aufgebaut, wie ich ihn tatsächlich aufsetzen würde...Ich glaube, das hat auch einfach was damit zu tun: Wie groß ist der Laden?...*"

Ein weiterer Teilnehmer stellte die Länge der Informationsflüsse innerhalb großer Unternehmen als ein mögliches Problem dar:

> "*ID18: Den wenden wir...an, und ich finde den auch gut. Das was bei uns hakt sind die Methodiken. [In] der Anforderungsanalyse kommt...die Verbindung zu Usergruppen [zu kurz]. Meistens [sprechen wir] mit dem Kunden und nicht [mit] den Usern direkt. Diesen Informationsfluss wieder zurückzuführen [durch die Abteilungen zu den Entwicklern]...das ist zu verzögert, du brauchst das Feedback sofort, damit du darauf reagieren kannst...*"

6.3 Ergebnis

Die Ergebnisse aus den Interviews überschneiden sich mit dem Workflow auf Basis des Literaturreviews. Alle Teilnehmer legten großen Wert auf endnutzerzentrierte Methoden innerhalb der Anforderungsanalyse und der Evaluation. Das Anforderungsmanagement und die Kodierung wurden für die Entwicklung erklärbarer Systeme im Vergleich eher nicht betrachtet. Innerhalb der Aktivität *Entwurf* wurden insbesondere Low-fidelity Prototypen wie Mock-Ups oder Papier-Prototypen empfohlen. In der *Anforderungsanalyse* wurde auch zusätzlich das Problem des Trade-offs mit anderen Anforderungen bestätigt und sollte daher betrachtet werden. Innerhalb der *Evaluation* fokussierten sich die Teilnehmer auf Usability Tests, insebsondere auch daher, da die Erklärbarkeit als ein Aspekt der Benutzerfreundlichkeit und -erfahrung angesehen wurde. Auch die Endnutzerobservierung wurde häufig erwähnt. Dadurch kann das Verständnis und die Reaktion des Endnutzers, wie beispielsweise Verwunderungen oder Überraschungen, die während der Systemanwendung auftreten können, erfasst werden. Mit Umfragen oder Interviews kann anschließend das Verständnis auf Korrektheit überprüft werden, was mit dem mentalen Modell aus der Literatur gleichgesetzt werden kann. Insgesamt haben alle Teilnehmer zugestimmt, dass der endnutzerzentrierte, iterative Entwicklungsprozess mit einer Evaluationsaktivität essenziell sei um erklärbare Systeme zu entwickeln. Dabei sollten bestenfalls qualitative, wie auch quantitative Methoden angewendet werden.

Da agile Vorgehensmodelle viele dieser Eigenschaften abdecken, sind diese gut für die Entwicklung erklärbarer Systeme geeignet. Da insbesondere alle Teilnehmer einen agilen Entwicklungsprozess im Unternehmen verfolgen, und sich die aktuell verwendeten Methoden mit den Methoden aus dem Workflow teilweise überschneiden, ist die Hürde für die Entwicklung erklärbarer Systeme gering. Lediglich der Informationsfluss des Feedbacks innerhalb der Unternehmen bereitet einigen Teilnehmern sorgen. Hierfür braucht es innovative Lösungen.

Kapitel 7

Zusammenfassung und Ausblick

Dieses Kapitel stellt einen abschließenden Überblick über die Inhalte und die wichtigsten Ergebnisse der Arbeit dar.

7.1 Zusammenfassung

Erklärbare Systeme werden durch die zunehmend komplexen Algorithmen immer wichtiger. Beispielsweise wird an der Entwicklung autonomer Fahrzeuge geforscht, mit der Vision eines vollends sich selbst steuernden Autos, welches Gefahren erkennt und angemessen reagiert. Diese Vision kann einigen Endnutzern sorgen bereiten, da sie befürchten dem autonomen Fahrzeug nicht vertrauen zu können. Damit insbesondere die zukünftigen Softwaresysteme den Alltag in einer positiven Art und Weise gestalten können, ist die Erklärbarkeit eine wichtige Anforderung. Sie ermöglicht den Endnutzer die Systemausgaben und -verhalten unter anderem zu verstehen, zu überprüfen oder zu vertrauen. Hier setzt der Workflow für erklärbare Systeme an, um eine im industriellen Kontext anwendbare Vorgehensweise für die Entwicklung erklärbarer Systeme zu bieten. Der Workflow wurde mit einem Literaturreview erstellt. Sie bildet die notwendigen Aktivitäten und Methodiken aus dem Requirements Engineering und der Mensch-Maschine Kommunikation ab. Anschließend wurde der Workflow mit Experteninterview evaluiert, indem er mit dem aktuellen Vorgehen im Unternehmen, sowie mit einem vom Experten selbstständig erstellten Workflows für erklärbare Systeme verglichen wurde. Die Ergebnisse aus der wissenschaftlichen Literatur und der Experteninterviews überschneiden sich. Aus beiden Ansätzen geht hervor, dass sich ein endnutzerzentrierter und iterativer Softwareentwicklungsprozess für die Entwicklung erklärbarer Softwaresysteme eignet. Da insbesondere agile Vorgehensmodelle diese Eigenschaften aufweisen, wird von den Interviewteilnehmern ein agiler

Softwareentwicklungsprozess wie Scrum oder SAFe bevorzugt erwähnt.

Forschungsfrage 1: Weiterhin wurden in beiden Ansätzen endnutzerzentrierte Methoden und Aktivitäten für die Entwicklung erklärbarer Software empfohlen. Da Erklärungen stark vom Wissen und dem mentalen Modell eines Endnutzers abhängen, ist die Analyse des Endnutzers und die Evaluierung des Systems durch den Endnutzer für die Erklärbarkeit essenziell. Das System kommuniziert anhand der Erklärungen mit dem Endnutzer, weshalb sich insbesondere Methoden aus dem Fachgebiet der Mensch-Maschine Kommunikation eignen, welches sich intensiv mit dieser Thematik beschäftigt. Weiterhin eignen sich Methoden des Requirements Engineerings, welche die Erfassung der Anforderungen der Stakeholder, ihre Bedürfnisse und Ziele sowie den Anwendungskontext anstreben.

Forschungsfrage 2: Mit den Ergebnissen aus der ersten Forschungsfrage wurde ein Workflow für die Entwicklung von erklärbaren Softwaresystemen aufgestellt. Unterstützend wurden verwandte Arbeiten herangezogen, in denen ein Prozess für die Entwicklung von erklärbaren Systemen entwickelt wurde. Dieser auf der wissenschaftlichen Literatur basierende Workflow wurde auf Eignung für die Realität anhand von Experteninterviews überprüft. Hierfür haben die Experten innerhalb des Interviews einen Workflow für erklärbare Systeme in einem kollaborativen Online-Zeichentool erstellt. Diese Workflows wurden anschließend mit dem auf der Literatur basierenden Workflows gegenübergestellt. Die Workflows aus den Interviews bestätigen den literaturbasierten Workflow in der realen Anwendbarkeit. Die verwendeten Methoden und Aktivitäten überschneiden sich in großem Maße. Alle Experten versichern dazu, dass insbesondere endnutzerzentrierte Methoden innerhalb der Anforderungsanalyse und der Evaluation essenziell sind, um wirksame erklärbare Systeme zu Entwickeln.

Forschungsfrage 3: Mögliche Einschränkungen oder Probleme, die bei der Einführung eines erklärbaren Workflows im Unternehmenskontext auftreten könnten, wurden in den Interviews angesprochen. In erster Linie wurde das Mindset der durchführenden Personen angesprochen, welche die Notwendigkeit der Erklärbarkeit und ihre Vorteile kennen, kommunizieren und leben müssen. Ohne das Bewusstsein würde selbst in einem endnutzerzentrierten Entwicklungsprozess kein optimales erklärbares System entstehen können. Insbesondere Interviews, Umfragen und Endnutzerobservierungen müssen zur Evaluierung der Ziele der Erklärbarkeit ausgerichtet werden, um geeignete Informationen aus diesen Methoden herauszuziehen. Bezüglich Problematiken in Ressourcen des Unternehmens wie der Zeitaufwand, monetäre Kosten oder die Mitarbeiteranzahl wurden von keinem der Teilnehmer Probleme oder Einschränkungen erwähnt. Der endnutzerzentrierte Workflow

mit der Ausrichtung für erklärbare Systeme sei realitätstauglich. Dieser müsse jedoch angewendet werden, um gegebenenfalls auftretende Probleme während der Durchführung aufzudecken und sie anschließend zu lösen.

7.2 Limitation

Viele wissenschaftliche Arbeiten beschäftigen sich derzeit mit der Entwicklung von erklärbaren Systemen. Während des Literaturreviews ist aufgefallen, dass die Mehrheit jedoch keine Details darüber nennt, mit welchen Aktivitäten und Methoden diese erklärbaren Systeme entworfen und entwickelt wurden. Mit derselbigen Problematik konfrontiert, empfiehlt Müller [77] daher Forscherinnen und Forscher ihre Herangehensweise genau zu erläutern. Eine detaillierte Beschreibung würde der Analyse und Forschung von erklärbaren Systemen mehr Einsicht und Grundlage bieten. Da in dieser Arbeit die derzeitig genannten Methoden und Aktivitäten gesammelt und analysiert wurden, ist der Workflow eher ein Abbild der aktuellen Forschung und kann daher nicht als vollkommen abgeschlossen verstanden werden. Darüber hinaus gab es in den Interviews einen Zeitrahmen wodurch es möglich ist, dass einige Methoden, Aktivitäten und andere Aspekte nicht erwähnt wurden. Auch kann ein Bias aufgetreten sein, da beispielhafte Methoden und Aktivitäten innerhalb der Aufgabenstellung aufgelistet wurden, die den Teilnehmer in seiner Antwort beeinflusst haben könnte.

7.3 Ausblick

Trotz dessen, dass in den Interviews die Notwendigkeit und Durchführbarkeit des Workflows bestätigt wurde, muss der Workflow in einem realen Kontext angewendet und getestet werden, um genaue und belegbare Aussagen treffen zu können. Außerdem sollte der Workflow mit den Ergebnissen neuer Forschungsergebnisse zukünftig fortlaufend erweitert werden. Weiterhin ist die Analyse einzelner Methoden bezüglich der Erklärbarkeit interessant. Beispielsweise können User Storys oder Use Cases verwendet werden, um einen Erklärungsbedarf für den Endnutzer festzustellen. Andererseits könnte auch die Notwendigkeit einer Erklärung in diesen festgehalten werden, wodurch sie als eine Kommunikationsform der Erklärbarkeitsanforderung dienen kann.

Anhang A

Anhang

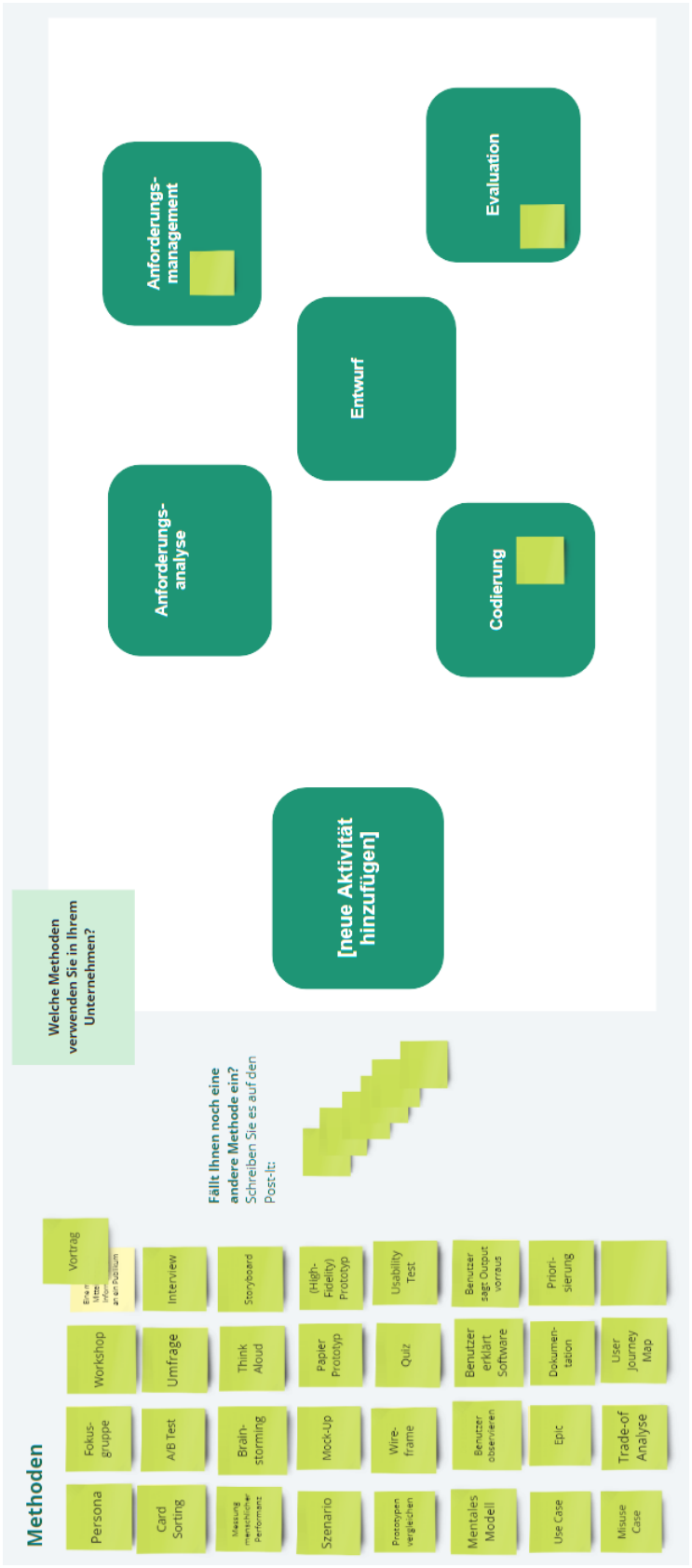

Abbildung A.1: Definition und Darstellung der Problematik anhand eines Szenarios

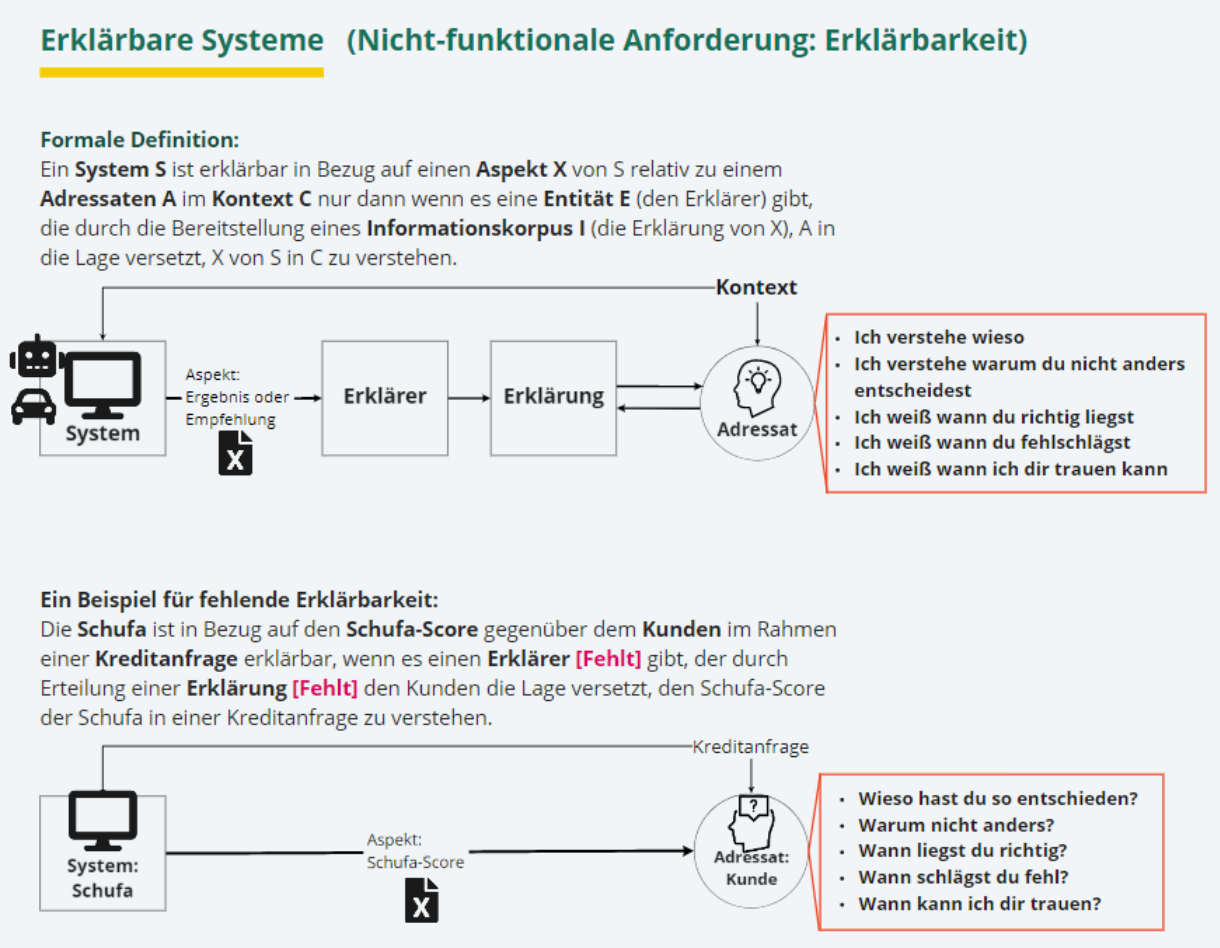

Abbildung A.2: Definition und Darstellung der Problematik anhand eines Szenarios

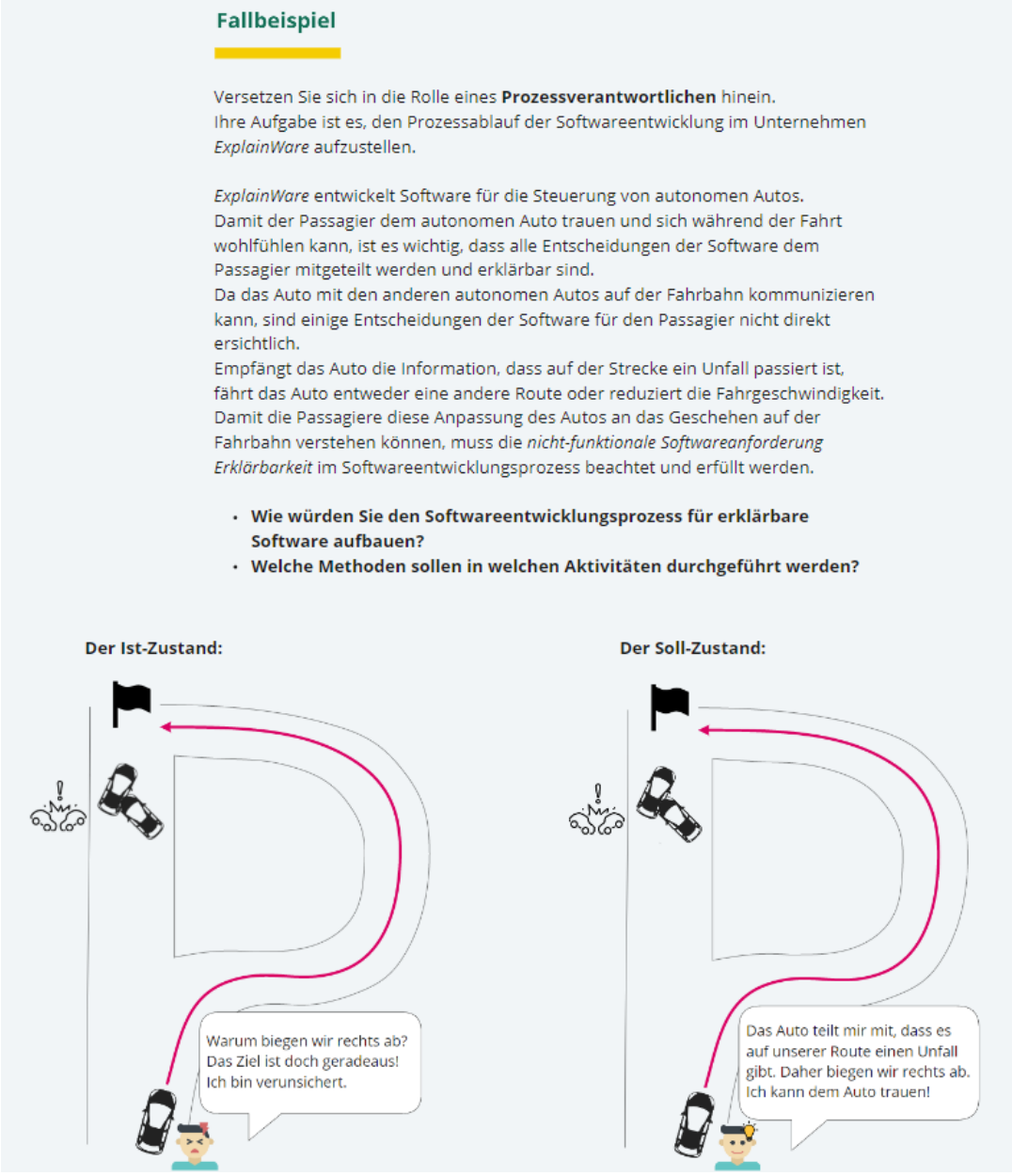

Abbildung A.3: Definition und Darstellung der Problematik anhand eines Szenarios

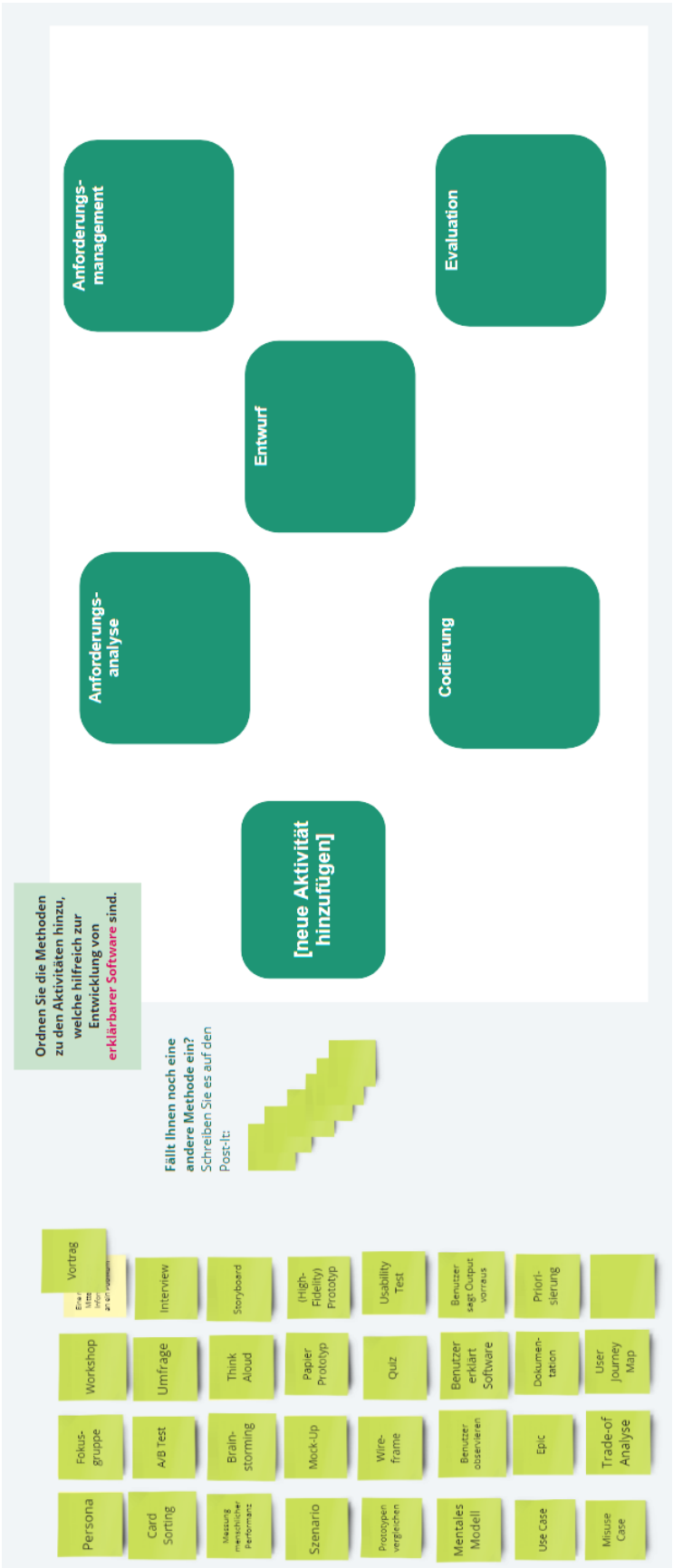

Abbildung A.4: Definition und Darstellung der Problematik anhand eines Szenarios

Literaturverzeichnis

[1] *DIN EN ISO 9241-210 Ergonomie der Mensch-System-Interaktion – Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme.* Berlin: Beuth, 2012.

[2] D. Ameller, C. Ayala, X. Franch, and J. Cabot. How do software architects consider non-functional requirements: An exploratory study. 09 2012.

[3] J. Andres, C. T. Wolf, S. Cabrero Barros, E. Oduor, R. Nair, A. Kjærum, A. B. Tharsgaard, and B. S. Madsen. Scenario-based xai for humanitarian aid forecasting. page 1–8, 2020.

[4] O. Anuyah, W. Fine, and R. Metoyer. Design decision framework for ai explanations. In C. Wienrich, P. Wintersberger, and B. Weyers, editors, *Mensch und Computer 2021 - Workshopband*, Bonn, 2021. Gesellschaft für Informatik e.V.

[5] V. Arya, R. K. E. Bellamy, P. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilovic, S. Mourad, P. Pedemonte, R. Raghavendra, J. T. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. *CoRR*, abs/1909.03012, 2019.

[6] V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilovic, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. Ai explainability 360: Impact and design, 2021.

[7] H. Balzert. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering, Lehrbücher der Informatik.* Spektrum Akademischer Verlag;, Heidelberg, 2009.

[8] O. Barkan, Y. Fuchs, A. Caciularu, and N. Koenigstein. Explainable recommendations via attentive multi-persona collaborative filtering, 09 2020.

[9] V. Belle and I. Papantonis. Principles and practice of explainable machine learning. *Frontiers in Big Data*, 4:39, 2021.

[10] U. Bhatt, M. Andrus, A. Weller, and A. Xiang. Machine learning explainability for external stakeholders. *CoRR*, abs/2007.05408, 2020.

[11] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery.

[12] A. Bibal, M. Lognoul, A. de Streel, and B. Frénay. Legal requirements on explainability in machine learning. *Artificial Intelligence and Law*, 29(2):149–169, 2021.

[13] M. Bilgic and R. Mooney. Explaining recommendations: Satisfaction vs. promotion. 01 2005.

[14] A. Boland, G. Cherry, and R. Dickson. Doing a systematic review: A student's guide. 2017.

[15] R. R. Bond, M. D. Mulvenna, H. Wan, D. D. Finlay, A. Wong, A. Koene, R. Brisk, J. Boger, and T. Adel. Human centered artificial intelligence: Weaving ux into algorithmic decision making. In *RoCHI*, pages 2–9, 2019.

[16] E. Börger, B. Hörger, D. Parnas, and H. Rombach. Requirements capture, documentation, and validation. In *Dagstuhl Seminar*, number 99241. Citeseer, 1999.

[17] R. Brenner and S. Wunder. Scaled agile framework: Presentation and real world example. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 1–2, 2015.

[18] A. Bunt, M. Lount, and C. Lauzon. Are explanations always important? a study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, page 169–178, New York, NY, USA, 2012. Association for Computing Machinery.

[19] A. Bussone, S. Stumpf, and D. O'Sullivan. The role of explanations on trust and reliance in clinical decision support systems. In *2015 International Conference on Healthcare Informatics*, pages 160–169, 2015.

[20] F. Cech and M. Wagner. Erollin' on green: A case study on eco-feedback tools for emobility. In *Proceedings of the 9th International Conference on Communities; Technologies - Transforming Communities*, page 121–125, New York, NY, USA, 2019. Association for Computing Machinery.

[21] T. Chakraborti, S. Sreedharan, and S. Kambhampati. The emerging landscape of explainable AI planning and decision making. *CoRR*, abs/2002.11697, 2020.

[22] P. Chakraborty, B. C. Kwon, S. Dey, A. Dhurandhar, D. Gruen, K. Ng, D. Sow, and K. R. Varshney. Tutorial on human-centered explainability for healthcare. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, KDD '20, page 3547–3548, New York, NY, USA, 2020. Association for Computing Machinery.

[23] S. Chari, P. Chakraborty, M. Ghalwash, O. Seneviratne, E. K. Eyigöz, D. M. Gruen, F. S. Saiz, C. Chen, P. M. Rojas, and D. L. McGuinness. Leveraging clinical context for user-centered explainability: A diabetes use case. *CoRR*, abs/2107.02359, 2021.

[24] L. Chazette, W. Brunotte, and T. Speith. Exploring explainability: A definition, a model, and a knowledge catalogue. *CoRR*, abs/2108.03012, 2021.

[25] L. Chazette and K. Schneider. Explainability as a non-functional requirement: challenges and recommendations. *Requirements Engineering*, 25(4):493–514, 2020.

[26] H.-F. Cheng, R. Wang, Z. Zhang, F. O'Connell, T. Gray, F. M. Harper, and H. Zhu. Explaining decision-making algorithms through ui: Strategies to help non-expert stakeholders. CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.

[27] L. Christino, M. D. Ferreira, A. Jalilvand, and F. V. Paulovich. Explainable patterns: Going from findings to insights to support data analytics democratization. *CoRR*, abs/2101.08655, 2021.

[28] M. Chromik and A. Butz. Human-xai interaction: A review and design principles for explanation user interfaces. In C. Ardito, R. Lanzilotti, A. Malizia, H. Petrie, A. Piccinno, G. Desolda, and K. Inkpen, editors, *Human-Computer Interaction – INTERACT 2021*, pages 619–640, Cham, 2021. Springer International Publishing.

[29] M. Chromik and M. Schuessler. A taxonomy for human subject evaluation of black-box explanations in xai. In *ExSS-ATEC@IUI*, 2020.

[30] D. Cirqueira, D. Nedbal, M. Helfert, and M. Bezbradica. Scenario-based requirements elicitation for user-centric explainable ai. In A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 321–341, Cham, 2020. Springer International Publishing.

[31] K. Cyras, R. Badrinath, S. K. Mohalik, A. Mujumdar, A. Nikou, A. Previti, V. Sundararajan, and A. V. Feljan. Machine reasoning explainability. *CoRR*, abs/2009.00418, 2020.

[32] H. K. Dam, T. Tran, and A. Ghose. Explainable software analytics. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ICSE-NIER '18, page 53–56, New York, NY, USA, 2018. Association for Computing Machinery.

[33] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey, 2020.

[34] D. Doran, S. Schulz, and T. R. Besold. What does explainable AI really mean? A new conceptualization of perspectives. *CoRR*, abs/1710.00794, 2017.

[35] F. Du, C. Plaisant, N. Spring, K. Crowley, and B. Shneiderman. Eventaction: A visual analytics approach to explainable recommendation for event sequences. 9(4), Aug. 2019.

[36] U. Ehsan, Q. V. Liao, M. J. Muller, M. O. Riedl, and J. D. Weisz. Expanding explainability: Towards social transparency in AI systems. *CoRR*, abs/2101.04719, 2021.

[37] U. Ehsan and M. O. Riedl. Human-centered explainable ai: Towards a reflective sociotechnical approach. In C. Stephanidis, M. Kurosu, H. Degen, and L. Reinerman-Jones, editors, *HCI International 2020 - Late Breaking Papers: Multimodality and Intelligence*, pages 449–466, Cham, 2020. Springer International Publishing.

[38] M. Eiband, H. Schneider, M. Bilandzic, J. Fazekas-Con, M. Haug, and H. Hussmann. Bringing transparency design into practice. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, page 211–223, New York, NY, USA, 2018. Association for Computing Machinery.

[39] X. Ferre, N. Juristo, and A. M. Moreno. Improving software engineering practice with hci aspects. In C. V. Ramamoorthy, R. Lee, and K. W. Lee, editors, *Software Engineering Research and Applications*, pages 349–363, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[40] J. J. Ferreira and M. de Souza Monteiro. Designer-user communication for XAI: an epistemological approach to discuss XAI design. 2021.

[41] J. J. Ferreira and M. S. Monteiro. What are people doing about xai user experience? a survey on ai explainability research and practice. In A. Marcus and E. Rosenzweig, editors, *Design, User Experience, and Usability. Design for Contemporary Interactive Environments*, pages 56–73, Cham, 2020. Springer International Publishing.

[42] B. Ghai, Q. V. Liao, Y. Zhang, R. Bellamy, and K. Mueller. Explainable active learning (xal): Toward ai explanations as interfaces for machine teachers. 4(CSCW3), Jan. 2021.

[43] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *CoRR*, abs/1806.00069, 2018.

[44] A. I. Grimaldo and J. Novak. User-centered visual analytics approach for interactive and explainable energy demand analysis in prosumer scenarios. In D. Tzovaras, D. Giakoumis, M. Vincze, and A. Argyros, editors, *Computer Vision Systems*, pages 700–710, Cham, 2019. Springer International Publishing.

[45] D. Gunning and D. Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019.

[46] M. Hall, D. Harborne, R. Tomsett, V. Galetic, S. Quintana-Amate, A. Nottle, and A. Preece. A systematic method to understand requirements for explainable ai (xai) systems. 2019.

[47] X. He, T. Chen, M.-Y. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. CIKM '15, page 1661–1670, New York, NY, USA, 2015. Association for Computing Machinery.

[48] J. Heier. Design intelligence - taking further steps towards new methods and tools for designing in the age of ai. In H. Degen and S. Ntoa, editors, *Artificial Intelligence in HCI*, pages 202–215, Cham, 2021. Springer International Publishing.

[49] L.-V. Herm, J. Wanner, F. Seubert, and C. Janiesch. I don't get it, but it seems valid! the connection between explainability and comprehensibility in (x) ai research. 2021.

[50] M. Hind. Explaining explainable ai. *XRDS*, 25(3):16–19, Apr. 2019.

[51] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Metrics for explainable AI: challenges and prospects. *CoRR*, abs/1812.04608, 2018.

[52] E. Holder and N. Wang. Explainable artificial intelligence (xai) interactively working with humans as a junior cyber analyst. *Human-Intelligent Systems Integration*, 3(2):139–153, 2021.

[53] A. Holzinger, A. M. Carrington, and H. Müller. Measuring the quality of explanations: The system causability scale (SCS). comparing human and machine explanations. *CoRR*, abs/1912.09024, 2019.

[54] M. N. Hoque and K. Mueller. Outcome-explorer: A causality guided interactive visual interface for interpretable algorithmic decision making. *CoRR*, abs/2101.00633, 2021.

[55] F. Hussain, R. Hussain, and E. Hossain. Explainable artificial intelligence (XAI): an engineering perspective. *CoRR*, abs/2101.03613, 2021.

[56] S. Jesus, C. Belém, V. Balayan, J. a. Bento, P. Saleiro, P. Bizarro, and J. a. Gama. How can i choose an explainer? an application-grounded evaluation of post-hoc explanations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 805–815, New York, NY, USA, 2021. Association for Computing Machinery.

[57] W. Jin, J. Fan, D. Gromala, P. Pasquier, and G. Hamarneh. EUCA: A practical prototyping framework towards end-user-centered explainable artificial intelligence. *CoRR*, abs/2102.02437, 2021.

[58] P. T. A. Junior and L. V. L. Filgueiras. User modeling with personas. In *Proceedings of the 2005 Latin American Conference on Human-Computer Interaction*, CLIHC '05, page 277–282, New York, NY, USA, 2005. Association for Computing Machinery.

[59] E. Kallina. Delegating agency? the effects of xai, personality traits, and the moral significance of the application on the reliance on autonomous systems: A user study.

[60] S. P. Karahasanović A., Følstad A. Putting a face on algorithms: Personas for modeling artificial intelligence. In H. Degen and S. Ntoa, editors, *Artificial Intelligence in HCI*, pages 202–215, Cham, 2021. Springer International Publishing.

[61] M. A. Köhl, K. Baum, M. Langer, D. Oster, T. Speith, and D. Bohlender. Explainability as a non-functional requirement. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 363–368, 2019.

[62] B. Kim, H. Suh, J. Heo, and Y. Choi. Ai-driven interface design for intelligent tutoring system improves student engagement. *CoRR*, abs/2009.08976, 2020.

[63] R. Kohavi and R. Longbotham. *Online Controlled Experiments and A/B Testing*. 01 2017.

[64] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, page 126–137, New York, NY, USA, 2015. Association for Computing Machinery.

[65] T. Kulesza, S. Stumpf, M. Burnett, W.-K. Wong, Y. Riche, T. Moore, I. Oberst, A. Shinsel, and K. McIntosh. Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 41–48, 2010.

[66] M. Langer, D. Oster, T. Speith, H. Hermanns, L. Kästner, E. Schmidt, A. Sesing, and K. Baum. What do we want from explainable artificial intelligence (xai)? – a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research. *Artificial Intelligence*, 296:103473, 2021.

[67] Q. V. Liao, D. M. Gruen, and S. Miller. Questioning the AI: informing design practices for explainable AI user experiences. *CoRR*, abs/2001.02478, 2020.

[68] B. Lim, A. Dey, and D. Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. *Conference on Human Factors in Computing Systems - Proceedings*, 04 2009.

[69] A. London. Artificial intelligence and black-box medical decisions: Accuracy versus explainability. *The Hastings Center Report*, 49:15–21, 02 2019.

[70] K. Martin, A. Liret, N. Wiratunga, G. Owusu, and M. Kern. Evaluating explainability methods intended for multiple stakeholders. *KI - Künstliche Intelligenz*, 2021.

[71] P. Mayring. Qualitative content analysis: Demarcation, varieties, developments. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 20(3), Sep. 2019.

[72] D. Mishra, A. Mishra, and A. Yazici. Successful requirement elicitation by combining requirement engineering techniques. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 258–263, 2008.

[73] S. Mohseni, N. Zarei, and E. D. Ragan. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. 11(3–4), Aug. 2021.

[74] I. Morales-Ramirez and L. H. Alva-Martinez. Requirements analysis skills: How to train practitioners? In *2018 IEEE 8th International Workshop on Requirements Engineering Education and Training (REET)*, pages 24–29, 2018.

[75] Y. Mualla, A. Najjar, T. Kampik, I. Tchappi, S. Galland, and C. Nicolle. Towards explainability for a civilian UAV fleet management using an agent-based approach. *CoRR*, abs/1909.10090, 2019.

[76] Y. Mualla, I. H. Tchappi, A. Najjar, T. Kampik, S. Galland, and C. Nicolle. Human-agent explainability: An experimental case study on the filtering of explanations. In *ICAART (1)*, pages 378–385, 2020.

[77] S. T. Mueller. Explanation in human-ai systems: A literature meta-review synopsis of key ideas and publications and bibliography for explainable ai. *DARPA XAI Program*, 02 2019.

[78] M. Naiseh, D. Al-Thani, N. Jiang, and R. Ali. Explainable recommendation: when design meets trust calibration. *World Wide Web*, 08 2021.

[79] M.-L. Nguyen, T. Phung, D.-H. Ly, and L. Truong. Holistic explainability requirements for end-to-end machine learning in iot cloud systems. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, United States, 2021. IEEE.

[80] M. Nourani, C. Roy, T. Rahman, E. D. Ragan, N. Ruozzi, and V. Gogate. Don't explain without verifying veracity: An evaluation of explainable AI with video activity recognition. *CoRR*, abs/2005.02335, 2020.

[81] D. Omeiza, K. Kollnig, H. Web, M. Jirotka, and L. Kunze. Why not explain? effects of explanations on human perceptions of autonomous driving. In *2021 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, pages 194–199, 2021.

[82] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. pages 308 – 313, 07 2003.

[83] H. Ramos, M. Fonseca, and L. Ponciano. Modeling and evaluating personas with software explainability requirements. *CoRR*, abs/2108.04640, 2021.

[84] J. Rebanal, J. Combitsis, Y. Tang, and X. A. Chen. Xalgo: A design probe of explaining algorithms' internal states via question-answering. In *26th International Conference on Intelligent User Interfaces*, IUI '21, page 329–339, New York, NY, USA, 2021. Association for Computing Machinery.

[85] M. Ribera Turró and A. Lapedriza. Can we do better explanations? a proposal of user-centered explainable ai. 03 2019.

[86] M. Richter and M. D. Flückiger. *Usability Engineering kompakt : Benutzbare Produkte gezielt entwickeln, IT kompakt.* Springer;, Berlin, 2013.

[87] K. Rob and D. Martin. *Code/space : software and everyday life.* The MIT Press, 2011.

[88] A. Rosenfeld and A. Richardson. Explainability in human-agent systems. *CoRR*, abs/1904.08123, 2019.

[89] C. Rupp. *Requirements Engineering - Ein Überblick.* SOPHIST GmbH, 2012.

[90] T. Schneider, J. Hois, A. Rosenstein, S. Ghellal, D. Theofanou-Fülbier, and A. R. Gerlicher. *ExplAIn Yourself! Transparency for Positive UX in Autonomous Driving.* Association for Computing Machinery, New York, NY, USA, 2021.

[91] T. A. Schoonderwoerd, W. Jorritsma, M. A. Neerincx, and K. van den Bosch. Human-centered xai: Developing design patterns for explanations of clinical decision support systems. *International Journal of Human-Computer Studies*, 154:102684, 2021.

[92] S. Schwarz and T. Roth-Berghofer. Towards goal elicitation by user observation. 2003.

[93] D. M. C. Seffah Ahmed, Vanderdonckt Jean. *Human-Centered Software Engineering - Software Engineering Models, Patterns and Architectures for HCI.* Springer, London, 2009.

[94] T. R. Society. *Explainable AI: the basics - Policy briefing.* The Royal Society, 2019.

[95] K. Sokol and P. Flach. Explainability fact sheets: A framework for systematic assessment of explainable approaches. In *FAT* 2020*

- *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pages 56–67, United States, Jan. 2020. Association for Computing Machinery (ACM). 3rd ACM Conference on Fairness, Accountability, and Transparency, FAT* 2020 ; Conference date: 27-01-2020 Through 30-01-2020.

[96] H. Subramonyam, C. M. Seifert, and E. Adar. Towards A process model for co-creating AI experiences. *CoRR*, abs/2104.07595, 2021.

[97] C. Tsai and P. Brusilovsky. Designing explanation interfaces for transparency and beyond. 02 2020.

[98] C.-H. Tsai and P. Brusilovsky. The effects of controllability and explainability in a social recommender system. *User Modeling and User-Adapted Interaction*, 31:591–627, 2021.

[99] C.-H. Tsai, Y. You, X. Gui, Y. Kou, and J. M. Carroll. *Exploring and Promoting Diagnostic Transparency and Explainability in Online Symptom Checkers*. Association for Computing Machinery, New York, NY, USA, 2021.

[100] T. Vermeire, T. Laugel, X. Renard, D. Martens, and M. Detyniecki. How to choose an explainability method? towards a methodical implementation of XAI in practice. *CoRR*, abs/2107.04427, 2021.

[101] G. Vilone and L. Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[102] D. Wang, Q. Yang, A. Abdul, and B. Y. Lim. Designing theory-driven user-centric explainable ai. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery.

[103] A. C. Weigand and M. C. Kindsmüller. Hcd3a: An hcd model to design data-driven apps. In H. Degen and S. Ntoa, editors, *Artificial Intelligence in HCI*, pages 285–297, Cham, 2021. Springer International Publishing.

[104] C. T. Wolf. Explainability scenarios: Towards scenario-based xai design. IUI '19, page 252–257, New York, NY, USA, 2019. Association for Computing Machinery.

[105] Y. Xie, M. Chen, D. Kao, G. Gao, and X. A. Chen. Chexplain: Enabling physicians to explore and understand data-driven, ai-enabled medical imaging analysis. In *Proceedings of the 2020 CHI Conference*

on Human Factors in Computing Systems, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[106] W. Xu. Toward human-centered ai: A perspective from human-computer interaction. *Interactions*, 26(4):42–46, June 2019.

[107] D. L. Yang L., Wang H. What does it mean to explain? a user-centered study on ai explainability. *Degen H., Ntoa S. (eds) Artificial Intelligence in HCI*, 02 2021.

[108] T. Zhou, H. Sheng, and I. Howley. Assessing post-hoc explainability of the bkt algorithm. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES 20, page 407–413, New York, NY, USA, 2020. Association for Computing Machinery.

[109] J. Zhu, A. Liapis, S. Risi, R. Bidarra, and G. M. Youngblood. Explainable ai for designers: A human-centered perspective on mixed-initiative co-creation. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2018.