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

Konzeption und Entwicklung eines Rahmenwerks zur Integration einzelner Werkzeuge für die multimediale Anforderungserhebung und -validierung

Design and Implementation of a Framework for Integrating Individual Tools for Multimedia Requirements Elicitation and Validation

Bachelorarbeit

im Studiengang Informatik

von

Christoph Hausmann

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M. Sc. Oliver Karras

Hannover, 08. September 2019

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit 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 08. September 2019

Christoph Hausmann

Zusammenfassung

Konzeption und Entwicklung eines Rahmenwerks zur Integration einzelner Werkzeuge für die multimediale Anforderungserhebung und -validierung

Im Requirements Engineering werden Anforderungen verschiedener Stakeholder aufgenommen. Um Missverständnisse zwischen dem Engineer und Stakeholder zu vermeiden, können unter anderem Vision Videos eingesetzt werden. Diese Videos eignen sich besser zur Kommunikation als schriftliche Dokumentationsarten, da Stakeholder in ihren eigenen Worten gehört werden können und die Videos genau das erfassen, was die Stakeholder gesagt haben. Jedoch werden Videos aus Gründen des vernommenen hohen Aufwands nur selten eingesetzt. In vorherigen Arbeiten wurden bereits Konzepte entwickelt, welche bei dieser Problematik helfen. Dabei haben sich diese Arbeiten nur auf einen Aspekt spezialisiert und für dessen Problematik einen Prototypen entwickelt.

In dieser Arbeit sollen diese Programme analysiert und in ein gemeinsames Rahmenwerk integriert werden, mit welchem der Benutzer ohne großen Aufwand Vision Videos aus Videos, Mockups und Bildern erstellen kann. Die ausgewählten Artefakte sollen dabei automatisch in Videos konvertiert und anschließend in ein einzelnes Gesamtvideo exportiert werden, welches zur Validierung an den Stakeholder gesendet werden kann. Das Ziel dabei ist, den Aufwand zur Erstellung von Vision Videos zu verringern und mithilfe von guter Usability Entwickler dazu anzuregen, Videos im Requirements Engineering häufiger in Betracht zu ziehen.

In einer Evaluation wurde ein beispielhaftes Szenario exemplarisch von den Probanden mithilfe des Prototypen visualisiert. Anschließend wurde das Design und die Bedienung mithilfe eines Fragebogens bewertet. Dabei wurde der Gesamteindruck und die Videoqualität eher neutral und die Bedienbarkeit mit Tendenz in das Gute bewertet.

Abstract

Design and Implementation of a Framework for Integrating Individual Tools for Multimedia Requirements Elicitation and Validation

One of the main aspects in requirements engineering is the collection of requirements. To combat misunderstandings between the engineer and stakeholder during the elicitation phase, vision videos can be used to help the communication. Since they have an on average higher effectiveness of communicating information than text, by allowing stakeholders to be heard in their own voice, videos would enrich this task. However, vision videos are rarely used in real practice. One of the main reasons for this is the perception of high effort creating them. Several papers have developed concepts in the past to combat this problem, however each of those only specialized in one feature.

In this work, the above mentioned concepts and programs are analyzed and afterwords integrated into a single joint framework that allows the user to easily create vision videos with little effort. The user should be able to use videos, mockups and images, which will be automatically converted into videos and then exported into a single video that contains all used artifacts. This video can be sent to the stakeholder for verification afterwords. The goal of this thesis is to decrease the effort for creating vision videos to increase the usage of videos in requirements engineering using high usability.

In an evaluation the prototype was tested by visualizing an exemplary scenario. The subjects then rated the design and usability of said prototype using a questionnaire. The subjects indicated a more neutral overall impression and video quality and a more positive tendency for the usability.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die technischen Entwicklungen in den letzten Jahrzehnten, vor allem in mobilen Geräten, erlauben es nicht nur schnell und überall erreichbar zu sein, sondern auch von unterwegs aus zu arbeiten. Solche Geräte, wie Tablets oder Notebooks, erlauben es dem Requirements Engineer, Kunden früher und stärker in die Entwicklung einzubinden, wodurch der Engineer schneller Feedback bekommt. Zudem erlauben mobile Geräte diesem, den Arbeitsplatz des Kunden zu besuchen, wodurch die Umwelt und Arbeitsweise des Kunden besser analysiert werden kann [3].

Im Requirements Engineering ist die Kommunikation und das gemeinsame Treffen mit dem Kunden ein wichtiges Ziel zum Erreichen eines gemeinsamen Verständnisses, vor allem in der agilen Software Entwicklung [6]. In diesen Treffen werden die Anforderungen des Kunden über das System erhoben und dokumentiert. Zudem werden Missverständnisse aufgeklärt und die Sichtweise sowie Wünsche aller Stakeholder ermittelt. Missverständnisse treten auf, da sich der Requirements Engineer nicht mit der Domäne des Kunden auskennt und dieser wiederum nicht die Techniken der Entwickler kennt. Diese sogenannte *Symmetry of Ignorance* [9] gilt es zu vermeiden, denn falsch beziehungsweise unzureichend erfasste Anforderungen sind ein häufiger Grund für das Scheitern eines Projekts und der Unzufriedenheit des Kunden [14].

Mobile Geräte können dabei aushelfen. Diese erlauben ein häufigeres Treffen auch vor Ort, da auch schwer erreichbare Kunden am Arbeitsplatz besucht werden können, sodass nicht auf ein Treffen am Arbeitsplatzes des Requirement Engineers gewartet werden muss [26]. Zudem können von mobilen Geräten aufgenommene Videos zur Dokumention der Anforderungen und Arbeitsweise des Kunden verwendet werden [3]. Diese haben einen höheren Informationsgehalt als die traditionell schriftliche Dokumentation und sind einfacher verständlich für die verschiedenen Kunden, da sie mit

ihrem eigenen Vokabular sprechen und gehört werden können. Dadurch kann eine höhere Anzahl an Teilnehmern in die Entwicklung miteinbezogen werden [3]. Wie schon in anderen Arbeiten [17], [25] gezeigt nehmen Videos verbale und nicht verbale Handlungen auf und lassen sich global einfach über das Internet austauschen. Nach Creighton et al. [7] sind Videos Repräsentationsformen, die alle Stakeholder verstehen können und Entwicklern eine reiche Basis an Referenzen bezüglich der Arbeitsweise des Kunden bieten. Mobile Geräte erlauben es auch dem Kunden, selbst Videos mit dem eigenen Tablet oder Smartphone aufzunehmen und ohne großen Aufwand an die Entwickler zu senden, wenn diesem neue Information eingefallen sind. Dabei helfen auch nicht professionell erstellte Videos fast genauso viel wie oder sogar mehr als die schriftliche Dokumentation [18]. Trotz dieser Vorteile werden Videos nur selten von den Engineers im Requirements Engineering genutzt, da sie häufig nicht das richtige Wissen für die Videoentwicklung besitzen oder ihnen der Aufwand zu groß erscheint [18].

In den letzten Jahren haben sich schon andere Arbeiten [11], [12], [27] mit dieser Thematik beschäftigt. Jedoch haben diese sich dabei meist nur auf einen bestimmten Aspekt spezialisiert und für diese Problemstellung passende Programme entwickelt. Alle sind dabei auf Ergebnisse gekommen, die für den Gebrauch von Videos im Requirements Engineering sprechen. Allerdings besteht das Problem, dass die Programme in ihrem Bereich hilfreich sind, jedoch nicht für den allgemeinen Gebrauch dienlich sind, da sie je nur eine Hauptfunktion bieten, wodurch der Aufwand gefühlt nicht reduziert wird. Diese Programme weisen gemeinsame Schnittstellen auf, die zueinander passen und sich unterstützen können. Es ist also wünschenswert, all diese Stärken in einer Software zu vereinigen, wodurch der Nutzer nur ein Programm benötigt, um Videos zu erstellen und editieren. Diese Software könnte den gefühlten Aufwand so reduzieren, dass Videos im Requirements Engineering häufiger in Betracht gezogen werden würden.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist die im vorherigen Abschnitt genannte Vereinigung der einzelnen Programme [11], [12], [27] in ein Gemeinsames. Dieses soll dann in der Anforderungserhebung und Dokumentation im Requirements Engineering behilflich sein, indem es dem Requirements Engineer mehrere Möglichkeiten der Videoproduktion bietet und ihn bei dieser unterstützt. Dadurch soll die Dokumentation verbessert und der Kunde nach dem von Brun-Cottan vorgestellten *Codevelopment* Konzept [3] mehr in die Softwareentwicklung mit eingebunden werden.

Besonders für die Entwicklung der Benutzeroberfläche und Interaktionsabläufe sollten die verschiedenen Nutzer miteinbezogen werden, damit die Oberflächen entsprechend der Bedürfnisse und Wünsche der Endnutzer ausgerichtet werden können [7], [19]. Die fertige Software soll es erlauben, nicht nur vorhandene Videos, Mockups und Bilder zu benutzen und vereinigen, sondern auch Neue zu erstellen und passend zu editieren. Diese Elemente sollen im Abschluss in ein gemeinsames Video exportiert werden können, welches mit dem Kunden geteilt oder als Referenzpunkt genutzt werden kann. Dabei soll auch ein späteres Editieren und Hinzufügen neuer Elemente möglich sein, um Anforderungen anzupassen. Wichtig dabei ist es den Aufwand für alle gering zu halten, um den Einsatz von Videos zu begünstigen, da nach einer Studie von Karras [16] der Aufwand von Videos als die mit Abstand größte Schwäche von Videos im Requirements Engineering unter den befragten Software Professionals angegeben wurde.

Dafür werden zunächst die Funktionen und Schnittstellen der einzelnen Programme ermittelt. Die Features, in denen sich die Programme spezialisiert haben, sollen so möglichst optimal miteinander verbunden und untereinander verwendet werden. Im weiteren Verlauf sollen dann die von den Autoren vorgeschlagenen Erweiterungen implementiert werden. Außerdem werden eigene Veränderungen und Erweiterungen an den Programmen an passenden Stellen vorgenommen.

Um nicht nur den Software Entwickler die Arbeit zu vereinfachen, sondern auch um den Kunden mit in die Entwicklung einzubeziehen, ist ein Hauptbestandteil dieser Arbeit der Fokus auf die User Experience. Das Programm soll bedienungsfreundlich und selbsterklärend sein, damit mehr Entwickler Videos als Option in Betracht ziehen und mit dem Kunden zusammen Mockups oder Skizzen der Oberflächen erstellen können.

Nach der Implementierung in ein Gesamtprozess soll dann die Software in einer Usability-Studie evaluiert werden, in der getestet wird, inwieweit das Design selbsterklärend ist und wie gut der Benutzer mit dem Programm arbeiten kann.

1.3 Struktur der Arbeit

In diesem Abschnitt wird kurz der Aufbau der Arbeit vorgestellt. Diese Arbeit gliedert sich in sieben Kapitel.

In Kapitel 1 soll die Motivation sowie das Ziel der Arbeit erläutert werden. Der Leser soll einen ersten Überblick über die Inhalte bekommen.

In Kapitel 2 werden die Grundlagen des Themas vermittelt, welche für das Verständnis der restlichen Arbeit notwendig sind. Dazu wird das Requirements Engineering und dessen Schritte erläutert. Anschließend werden die Begriffe Videos, Mockups und Szenarien speziell im Kontext des Requirements Engineering erläutert.

In Kapitel 3 werden die Funktionen und Implementierungen der Programme analysiert, welche im weiteren Verlauf in ein Gesamtkonzept verbunden werden sollen. Dabei werden diese Programme sowohl technisch als auch im Kontext der Nutzung miteinander verglichen.

In Kapitel 4 wird das Gesamtkonzept beschrieben. Dabei wird auf die Umwelt des Programms sowie dessen Funktionen und Anforderungen eingegangen. Zudem werden potenzielle Erweiterungen der Software beschrieben. Anschließend wird ein Fallbeispiel für den Gebrauch des Prototypen aufgezeigt und zuletzt Bezug auf verwandte Arbeiten genommen.

In Kapitel 5 wird die Implementierung des Programms vorgestellt sowie dessen Besonderheiten erläutert.

In Kapitel 6 wird das Gesamtkonzept auf Basis der Usability evaluiert. Dazu wird ein spezifiziertes Szenario von den Probanden visualisiert und die Bearbeitungszeit gemessen. Ebenfalls wird das Design mithilfe von Usability Heuristics evaluiert.

Im letzten Kapitel 7 werden die Ergebnisse schließlich zusammengefasst und ein Ausblick auf die Arbeit präsentiert.

Kapitel 2

Grundlagen

In diesem Kapitel werden einige Grundlagen vermittelt, die bei der späteren Verständnis der Arbeit helfen. Zuerst folgt die Definition des Requirements Engineering und deren einzelnen Phasen. Anschließend werden die Begriffe zu Videos, Mockups und Szenarien im Kontext des Requirements Engineering erläutert.

2.1 Requirements Engineering

Das Requirements Engineering (RE) ist ein wichtiger erster Schritt in der Softwareentwicklung und vom International Requirements Engineering Board (IREB) wie folgt definiert [4]:

Definition: Requirements Engineering

Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

1. Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen systematisch zu managen.
2. Die Wünsche und Bedürfnisse der Stakeholder zu verstehen und zu dokumentieren.
3. Die Anforderungen zu spezifizieren und zu managen, um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht.

Dieser Prozess besteht nach Börger [2], wie in Abbildung 2.1 zu sehen ist, aus zwei Teilprozessen, der *Requirements Analysis*, in welcher Anforderungen systematisch ermittelt werden, und dem *Requirements Management*, in welcher die Anforderungen verwaltet werden. Diese beiden Teilprozesse lassen sich daraufhin weiter aufteilen. Das systematische Requirements Engineering erhöht die Qualität des Produkts und die Produktivität der Entwickler ebenso wie die Reduktion der Kosten und nicht verwendeten Funktionen [8].

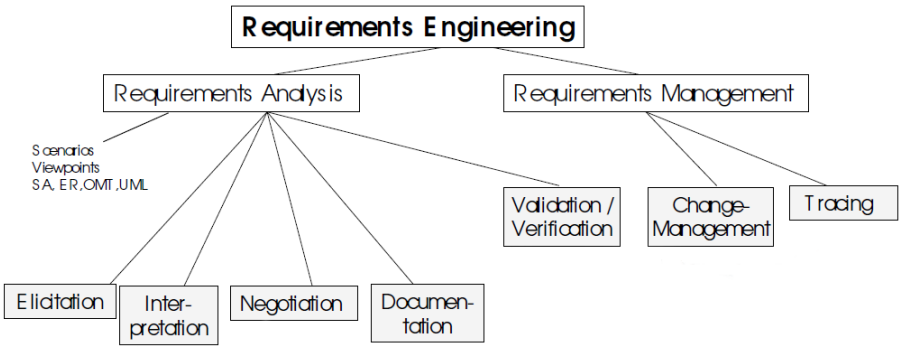

Abbildung 2.1: Requirements Engineering Referenz Modell ([2] S.30)

Die folgenden Aufführungen, sofern nicht anders angegeben, basieren auf Ebert [8], Pohl [21] und Schneider [24].

2.1.1 Requirements Analysis

Die Requirements Analysis (dt. Anforderungsanalyse) ist die systematische Ermittlung, Entwicklung und Analyse von Kundenanforderungen und der daraus abgeleiteten Anforderungen an das System [8]. Dieser Schritt dient dazu ein gemeinsames Verständnis zwischen den Kunden und Entwicklern herzustellen sowie mögliche Missverständnisse zu beseitigen. Dieser Schritt enthält die fünf Hauptaktivitäten, die im folgenden erläutert werden:

Elicitation

Die Elicitation-Phase ist der Anfang im Requirements Engineering und entspricht dem *Sammeln* im ersten Schritt der Abbildung 2.2. Es werden die Anforderungen für das Projekt gesammelt und verstanden. Dazu müssen zuerst alle Personen, die direkt oder indirekt von der Software betroffen sind oder diese beeinflussen, identifiziert werden. Diese sogenannten *Stakeholder* werden aufgesucht und deren Anforderungen durch ausgewählte Techniken, wie Interviews, Beobachtung der Arbeitsweise, Fragebögen oder Workshops, ermittelt. Eine weitere Quelle für Anforderungen ist das Umfeld des Systems, wozu Altsysteme, die Umweltbedingungen oder auch die Software vom

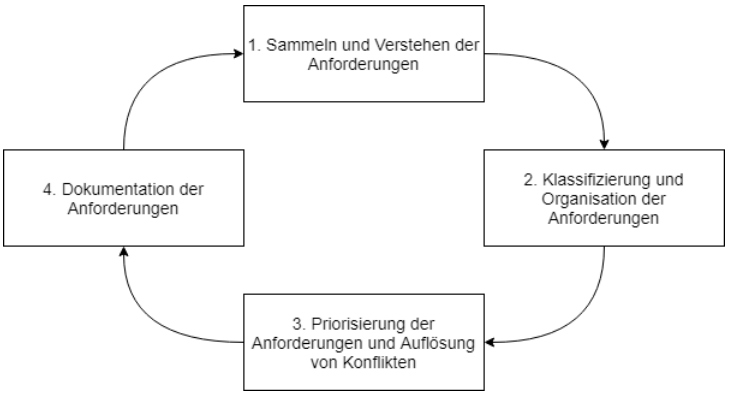

Abbildung 2.2: Der Prozess der Anforderungserhebung und -analyse ([28] S.133)

Konkurrenten gehören. So lassen sich Eindrücke und weitere Wünsche von den Kunden bestimmen, indem diese die anderen Programme benutzen und ihre Meinung dazu äußern. Zuletzt lassen sich auch aus der Dokumentation, wie universale Dokumente (zum Beispiel Standards) oder interne Dokumente (zum Beispiel gesammelte Anforderungen und Fehlermeldungen der Altsysteme), Anforderungen erfassen. Diese werden dabei häufig in *Use-Cases*, *StoryCards* oder im *Fließtext* dokumentiert [23].

Am Ende dieser Aktivität stehen die Rohanforderungen, die für spätere Zwecke noch ungeeignet sind, da sie noch unstrukturiert sind und lediglich die Ergebnisse der Befragungen enthalten. Diese gilt es im nächsten Schritt weiter zu bearbeiten.

Interpretation

In der Interpetation-Phase werden die aus der Elicitation-Phase erhobenen Rohanforderungen inhaltlich bearbeitet und im Kontext des Kunden interpretiert. Dazu werden diese inhaltlich identifiziert, strukturiert und konkretisiert. Die wichtigsten Informationen werden extrahiert und verfeinert. Um sie zu strukturieren, werden die Anforderungen organisiert und in Gruppen, wie funktionale und nicht-funktionale Anforderungen, eingeteilt. Diese helfen später bei der Priorisierung und Verteilung der Aufgaben auf das Team. Identische oder ähnliche Anforderungen werden dabei miteinander vereinigt und Beziehungen zwischen ihnen aufgezeigt. Diese Phase entspricht dem *Verstehen* der Anforderungen des ersten Schrittes sowie dem zweiten Schritt der Abbildung 2.2. Bei vielen Projektbeteiligten und der daraus resultierenden Menge an Anforderungen führt diese Aktivität zu Konflikten durch unterschiedliche Interessen oder Unklarheiten, die im nächsten Schritt aufgelöst werden müssen.

Negotiation

In der Negotiation-Phase werden die Abhängigkeiten der Anforderungen identifiziert und Konflikte aufgelöst. Diese entstehen durch verschiedene Perspektiven, die Widersprüche erschaffen, und Missverständnissen bei der Erhebung. Zur Lösung dieser müssen sich die beteiligten Stakeholder mit den Requirements Engineers treffen und Alternativen verhandeln, um eine einheitliche Basis zu bilden, die alle Betroffenen zufrieden stellt. Daraufhin werden die Anforderungen nach der Wichtigkeit priorisiert. Dies erlaubt es bei Zeitdruck oder Kostensteigung unwichtige Funktionen herauszunehmen, um den Kunden immer noch das gewünschte Produkt zu bieten.

Am Ende dieser Phase stehen die vom Kunden geforderten Anforderungen an das System, welche in der letzten Phase fixiert werden. Diese Phase entspricht dem dritten Schritt der Abbildung 2.2.

Documentation

In der Dokumentations-Phase werden die Anforderungen dokumentiert, um sowohl Basis und Referenzen zu haben als auch um die Anforderungen sortiert wieder zu finden. Dies findet meist parallel zu den anderen Phasen statt. Es werden nicht nur die Anforderungen fixiert, sondern auch die Gesprächspartner, deren Rollen und Ziele sowie die Hintergründe und Randbedingungen des Systems. Die Dokumentation geschieht im sogenannten Lasten- und Pflichtenheft, in denen alles schriftlich gesammelt und durch Diagramme, Tabellen und mögliche andere Zusatzanforderungen ergänzt wird. Am Ende stehen die Spezifikationen des Systems, die alle relevanten Informationen gesammelt fixieren. Bei Veränderungen müssen diese angepasst und erweitert werden.

Dies entspricht dem vierten Schritt in der Abbildung 2.2 und beginnt darauf folgend die nächste Iteration der Anforderungserhebung, sofern diese benötigt wird.

Validation/Verification

Im gesamten RE-Prozess muss geprüft werden, ob die Anforderungen diejenigen sind, die sich der Kunde auch wünscht. Dazu dient zum einen die Validation (dt. Validierung), in der überprüft wird, ob die dokumentierten Anforderungen auch die wirklichen Anforderungen sind. Dies wird inhaltlich geprüft durch Befragungen, Interviews oder Nachprüfung mit dem Kunden und geschieht zusammen mit dem vom Entwickler bereits erstellten Entwurf. Zum anderen wird durch die Verification (dt. Verifikation) formal geprüft, ob die Anforderungen des Entwurfs mit den zuvor dokumentierten

übereinstimmen. Dies geschieht durch Konsistenz- und Qualitätsprüfungen, in denen die Spezifikationen im Fehlerfall angepasst werden.

2.1.2 Requirements Management

Das Requirements Management (dt. Anforderungsmanagement) ist der Teil des Requirements Engineering, welcher sich mit der Evolution, Verwaltung und dem Management der Anforderungen im gesamten Entwicklungszeitraum beschäftigt [8]. Dies wird durch diverse Werkzeuge, wie zum Beispiel einfachen Datenbanksystemen oder speziellen Programmen, unterstützt.

Change Management

Das Change Management (dt. Änderungsmanagement) beschäftigt sich mit den Änderungen von Anforderungen während der Entwicklung. Anforderungen können sich im Verlauf des Projekts ändern, es werden sowohl Neue hinzugefügt als auch Alte verändert oder sogar entfernt. Gründe dafür sind neue Wünsche der Stakeholder oder unvollständigen alte Anforderungen. Vom Kunden gewünschte Änderungen sollen problemlos möglich und nachverfolgbar sein, um wenig Ressourcen in der Projektentwicklung zu beanspruchen. Dazu dienen Versionen von Anforderungen, die kontrolliert und angepasst werden. Im Change Management werden diese Änderungen analysiert und der Aufwand neu bewertet. Dies geschieht im sogenannten *Change Control Board*, welches ein für diesen Prozess speziell gebildetes Gremium ist. Danach wird entschieden, ob die Änderungen durchgesetzt werden und mit welcher Priorität, je nachdem wie das Gremium entscheidet. Alle Entscheidungen werden dabei dokumentiert, wobei wichtig ist auch den Grund der Änderung mit zu erfassen und die Änderungen weiter zu propagieren.

Tracing

Das Tracing (dt. die Verfolgbarkeit) ist die Fähigkeit, die Anforderung mit allen Veränderungen im gesamten Entwicklungsprozess verfolgen zu können. Es werden Beziehungen zwischen Anforderungen sowie deren Änderungen nachvollziehbar gezogen. Somit lässt sich überprüfen, ob die Anforderung bereits implementiert ist und erlaubt einfachere Instandhaltung der Software, indem sowohl der Grund und Effekt eines Fehlers gefunden werden kann als auch die betroffenen Komponenten einfacher identifiziert werden können. Dabei wird zwischen *pre-tracing*, das Festhalten der Annahmen und Anforderungen bis zu ihrer Entstehung in der Elicitation, und dem *post-tracing*, den Auswirkungen auf das System sowie den Abhängigkeiten zwischen den Systemkomponenten, unterschieden.

2.2 Videos im Requirements Engineering

Die im letzten Abschnitt 2.1 erläuterten Aktivitäten werden standardmäßig schriftlich fixiert. Viele Arbeiten [1], [10], [18] haben bereits nachgewiesen, dass dadurch Informationen verloren gehen. Laut Ambler [1] sind Videos die effektivsten Dokumentationsoption (siehe Abbildung 2.3). Auch Creighton [7] und Brun [3] sehen in Videos eine positive Ergänzung zur schriftlichen Dokumentation, da eine Videoaufnahme der realen Teilnahme an Interviews oder Workshops am nächsten kommen.

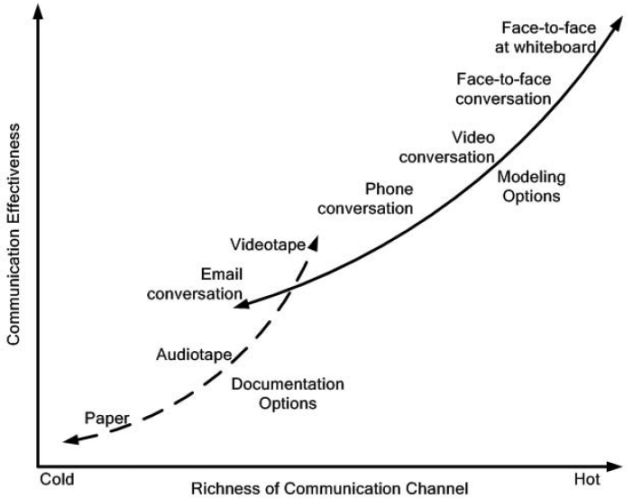

Abbildung 2.3: Modes of communication [1]

Videos im Kontext dieser Arbeit können zwar auch zur Dokumentation genutzt werden, werden hier aber für die Darstellung von Szenarien verwendet. Szenarien dienen der Systembeschreibung und repräsentieren einen erfolgreichen Anwendungsfall aus Sicht des Endnutzers sowie den Antworten des Systems. Die Videos können dann als Referenz für den Entwickler benutzt werden, da neben verbalen auch non-verbale Informationen mit aufgenommen werden [25]. Zudem können Videos benutzt werden, um bei Fragen und Ungewissheit die Anforderungen mit dem Kunden zu verifizieren (siehe 2.1). Karras und Schneider [18] haben gezeigt, das auch nicht professionell hergestellte Videos für diesen Zweck ausreichend sind.

2.3 Mockups

Das Mockup ist ein Bestandsteil des Prototyping, welches nach Ebert [8] definiert ist als:

Prototyping

Evolutionäre Vorgehensweise, um bei noch sehr unbestimmten Anforderungen schrittweise eine Lösung zu entwickeln (zum Beispiel Benutzungsschnittstellen). [...] Prototypen sollten weggeworfen werden, denn sie sind nicht auf Haltbarkeit und Wartbarkeit ausgelegt

Mockups speziell bieten hierfür die Möglichkeit, die graphische Benutzungsschnittstelle und deren Bedienungselemente darzustellen. Prototypen wie Mockups werden genutzt, um mit dem Stakeholder anhand einer verwendbaren Oberfläche Bedienungsabläufe durchzutesten und eventuelle Missverständnisse zu entdecken. Ein Mockup erlaubt es auch dem Entwickler, dem Stakeholder einen ersten Einblick in das Aussehen der Software zu bieten.

Prototypen lassen sich nach Houde and Hill [13] in drei Funktionsgruppen klassifizieren:

- **Role**
  Die Funktion, die für den späteren Benutzer erfüllt werden soll. Je früher diese mit dem Stakeholder getestet werden kann, desto mehr Zeit kann man beim Finden von unpassenden Lösungen sparen [7].
- **Implementation**
  Die Techniken und Komponenten, die verwendet werden, um die Funktion auszuführen.
- **Look and Feel**
  Das Aussehen und die sensorische Erfahrung, die der Benutzer während der Nutzung erfährt.

Es gibt verschiedene Formen von Prototypen, jede mit ihren eigenen Vor- und Nachteilen. Für diese Arbeit relevantesten sind das *Paper-Prototyping*, *Video-Prototyping* und *Facade-Prototyping*. *Paper Prototypes* sind Skizzen auf Papier und erlauben somit schnell und günstig erstes Feedback und Tests vom Benutzer. So lassen sich auch mit dem Kunden zusammen schnell Prototypen entwickeln, da dieser seine Wünsche selber skizzieren kann. Der große Nachteil ist, dass sich mit Paper-Prototypes nur schwer Interaktionen durchspielen lassen. Zur Vermittlung von Interaktionen eignen sich *Video Prototypes*, die durch ihren hohen Informationsgehalt (siehe Videos im Requirements Engineering) selbsterklärend die Interaktion und das Systemverhalten veranschaulichen. Die *Facade-Prototypes* ähneln bereits dem Aussehen von fertigen Anwendungen und erlauben in digitaler Form auch, Interaktionen durchzuspielen [22]. Somit lässt sich neben dem Design auch das Verhalten überprüfen.

Mockups sind dabei eine Form vom Facade-Prototyping, die Abbildung 2.4 zeigt so ein Mockup.

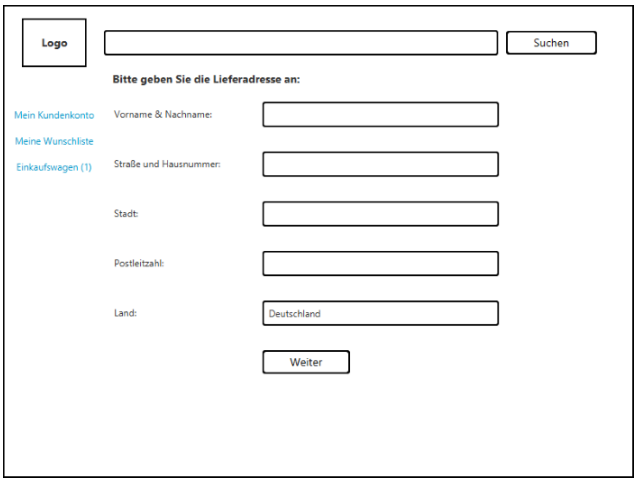

Abbildung 2.4: Beispiel eines Mockups zur Eingabe der Lieferadresse. Felder und Knöpfe sind anklickbar und ausfüllbar [11].

2.4 Szenarien

Es gibt verschiedene Definitionen von Szenarien. Pohl [21] definiert Szenarien wie folgt:

> **Szenario**
>
> Ein Szenario beschreibt ein konkretes Beispiel für die Erfüllung oder Nichterfüllung eines oder mehrerer Ziele. Es konkretisiert dadurch eines oder mehrere Ziele. Ein Szenario enthält typischerweise eine Folge von Interaktionsschritten und setzt diese in Bezug zum Systemkontext.

Diese sind im Kontext dieser Arbeit linear, um die Komplexität gering zu halten. Sie beschreiben ablauforientiert, wie sich Ereignisse oder Akteure entwickeln. Sie können die Ist-Situation beschreiben und mit *Visionary-Szenarien* können zukünftige Interaktionen beschrieben werden, wobei der Interaktionsfluss die Anforderungen beschreiben [5]. Durch Szenarien lassen sich auch Extremszenarien, wie das Best-Case- oder Worst-Case-Szenario, sowie besonders relevante Abläufe beschreiben. Dies ermöglicht einen leicht vorstellbaren Durchlauf, mit denen Abhängigkeiten und Anforderungen extrahiert werden können [8].

Kapitel 3

Softwareanalyse

In diesem Kapitel werden die Programme analysiert, die später in das Gesamtsystem integriert werden sollen. Dies sind der **Mockup Recorder** von Glauer [11] und der **Vision Video Recorder** von Happe [12]. Dazu werden diese Arbeiten zuerst kurz vorgestellt. Daraufhin werden die Hauptfunktionen jedes Programms sowie deren technischen Implementierungen ermittelt. Es folgen Schnittstellen der Tools und ein Vergleich beider Programme.

3.1 Einführung

Die hier vorgestellten Konzepte basieren auf den Informationen der jeweiligen Arbeiten.

Vision Video Recorder

Der **Vision Video Recorder** (kurz ViViRecorder) ist ein Tool zur Aufnahme von Vision Videos und dem Export dieser in ein Gesamtvideo. Dadurch lässt sich die Vorstellung des Systems leicht visualisieren und ermöglicht es die zuvor erhobenen Anforderungen vom Kunden validieren zu lassen. Das Ziel der Arbeit war es, ein Tool zu entwickeln, das den Nutzer erlaubt, ohne große Vorkenntnisse gute Videos zu erstellen. Um den Nutzer dabei zu unterstützen hat Happe [12] Video-Guidelines entwickelt, die während der Aufnahme automatisch ausgeführt werden und bei Einhaltung zu einer guten Videoqualität führen.

Mit dem Tool kann man Szenarien erstellen und für jeden Schritt des Szenarios entweder Videos aufnehmen oder bereits Vorhandene hinzufügen. Falls spätere Korrekturen nötig sind, lassen sich die Reihenfolge der Schritte sowie deren Namen und verwendeten Videos ändern.

Mockup Recorder

Der **Mockup Recorder** ist ein Tool zum Aufzeichnen und Abspielen von Interaktionsabläufen in Mockups sowie zum Erstellen und Editieren dieser. Die Mockups veranschaulichen die vom Kunden gewünschte Oberfläche, die zuvor von einem Designer entworfen wurde und mit dem Mockup Recorder nun getestet werden kann. Diese Interaktionen können dann in einem Video dokumentiert werden und dienen zur späteren Analyse. So lassen sich Anforderungen validieren und Feedback zu graphischen Oberflächen sammeln. Dafür hat Glauer [11] ein Konzept entwickelt, welches Mockups in das Programm lädt und sie so interagierbar macht. Dieses erlaubt auch, Fotos zu importieren, um sie in Mockups umzuwandeln.

In diesem Tool lassen sich ebenfalls Szenarien erstellen, wobei statt Videos Mockups verwendet werden. In den Szenarien können neue oder bereits vorhandene Mockups verwendet werden, die Schritte lassen sich ebenfalls editieren und die Reihenfolge ändern.

3.2 Konzepte

In diesem Abschnitt werden die grundlegenden Konzepte beider Programme sowie deren Hauptfunktionen ermittelt. Zudem wird das Design der einzelnen Programme präsentiert.

Vision Video Recorder

Die Grundidee ist, Szenarien in einem Video zu visualisieren und diese zur Verifikation der Anforderungen zu benutzen. Dazu hat der ViViRecorder ein Hauptfenster (siehe Abbildung 3.1), in dem der Livestream der Kamera zu sehen ist, welcher zur Aufnahme von Videos genutzt wird. Zudem besteht die Option, die Umweltumgebung des Geräts vor dem Start der Aufnahme zu analysieren, um ein bestmögliches Video zu erstellen. Für jedes zu erstellende Szenario lassen sich Schritte definieren und Videos hinzufügen. Für diese kann der Nutzer entweder mit dem Programm Neue aufnehmen oder bereits existierende Videos auf dem Gerät verwenden. Die Reihenfolge sowie Beschreibung der Schritte lässt sich beliebig ändern. Diese werden gespeichert und bleiben auch beim nächsten Aufruf erhalten.

Der Nutzer hat auch die Möglichkeit, in einem separaten Fenster die Videos im dafür integrierten Videoplayer (siehe Abbildung 3.2) hintereinander abzuspielen. Ist der Nutzer zufrieden mit den Videos, so kann er diese über einen Knopfdruck in ein gesamtes Video exportieren.

Zur Unterstützung des Nutzers dienen Video Guidelines, die auf Lichtverhältnisse, den Gerätewinkel, die unruhige Bewegung des Geräts sowie den Geräuschpegel achten.

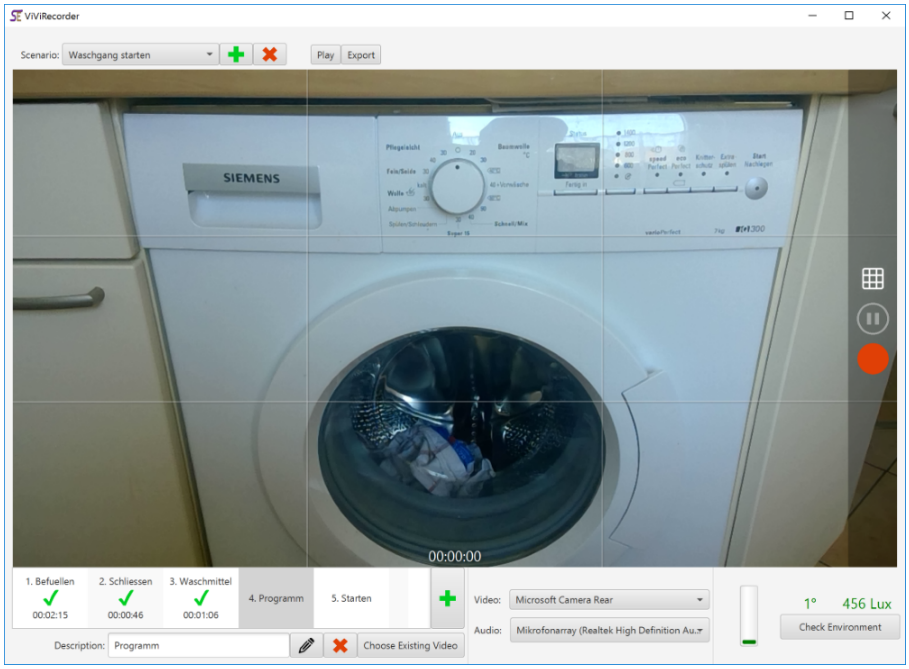

Abbildung 3.1: Hauptansicht des ViViRecorders [12]

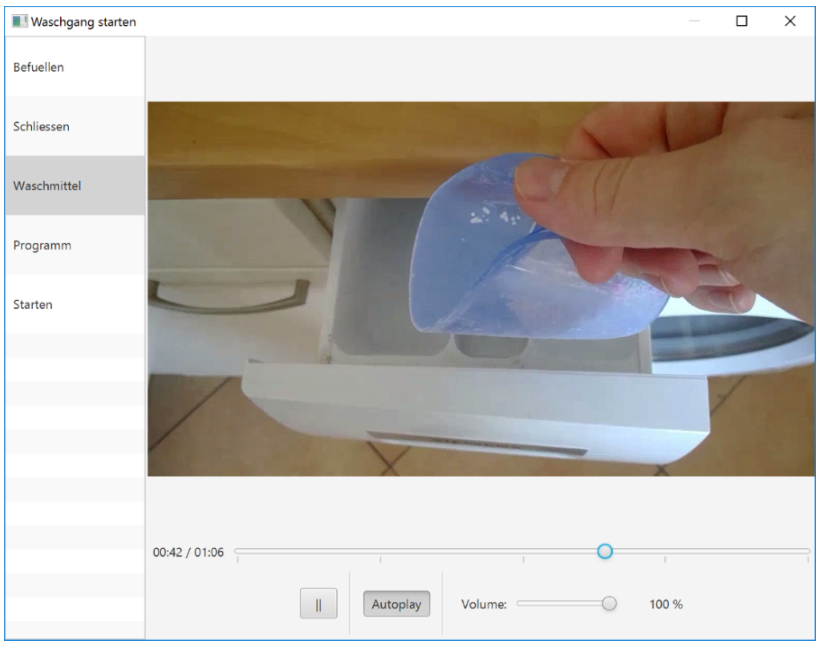

Abbildung 3.2: Videoplayer [12]

Der Benutzer kann vor der Aufnahme die Umwelt per Knopfdruck analysieren (siehe Abbildung 3.3). Bei Verstößen gegen eine oder mehrere Guidelines während der Aufnahme wird der Benutzer nach Aufnahmeende über den Fehler per Dialog informiert und die dazugehörige Stelle im Video in einer Logdatei zeitlich angeben. Damit wird der Nutzer während der Aufnahme nicht gestört und kann daraufhin selbst entscheiden, ob er das Video noch einmal aufnehmen will.

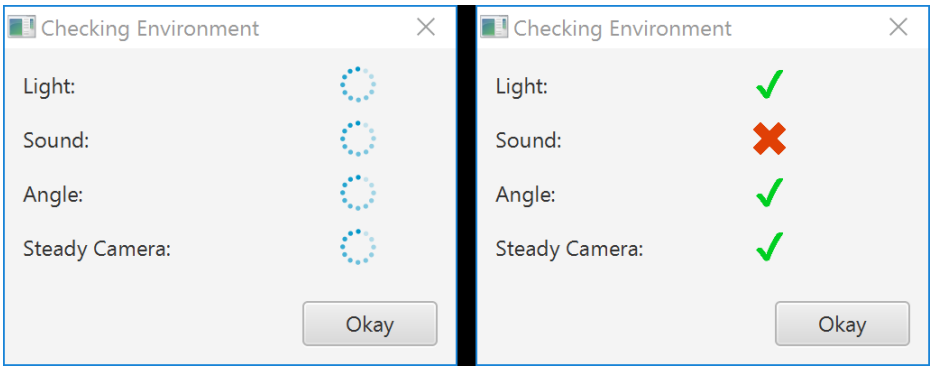

Abbildung 3.3: Umgebungsüberprüfung vor Aufnahme [12]

Der ViViRecorder erlaubt die Auswahl zwischen verschiedenen Video- und Audio-Inputs. Das Klicken auf den Record Button deaktiviert alle anderen Elemente, sodass bei der Aufnahme keine versehentliche Störung möglich ist. Auch lässt sich die Aufnahme per Knopfdruck pausieren. Der Nutzer kann ein Raster aktivieren, welches Anhaltspunkte für die Perspektive erlaubt.

Ein Zustandsdiagramm zum besseren Verständnis des ViViRecorders ist im Anhang (siehe A.2). Es stellt die Optionen, die der Benutzer bei der Erstellung von Videos hat, sowie das Systemverhalten dar.

Mockup Recorder

Die Grundidee hier ist es den Kunden zu erlauben, Interaktionen verschiedener Mockups zu testen und diese für den Entwickler aufzunehmen. Diese helfen bei der Erhebung, Dokumentation und Validierung von Anforderungen und erlauben ein schnelles Feedback der graphischen Oberfläche. Dazu existiert ein Hauptfenster (siehe Abbildung 3.4), in denen bereits erstellte Mockups vom Mockup Chooser (linke Seite) in die Mockup Timeline (untere Hälfte) gezogen werden können. Für jedes Mockup in der Timeline lassen sich daraufhin Interaktionen aufnehmen und wiedergeben. Auch hier repräsentieren diese Items Schritte eines Szenarios. Diese Interaktionen lassen sich zum Schluss entweder als Text, Bild oder Video exportieren.

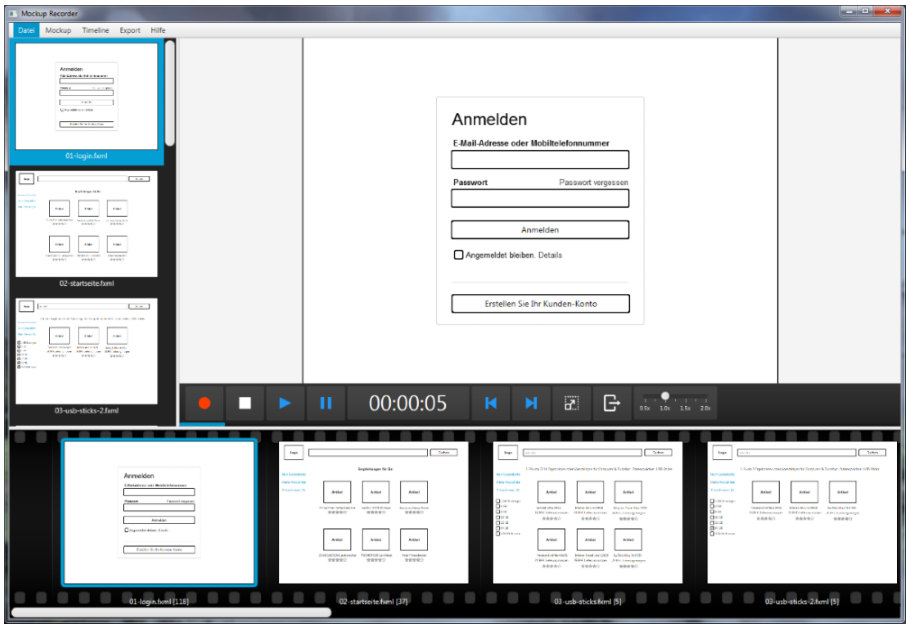

Abbildung 3.4: Hauptansicht des Mockup Recorders [11]

Es ist auch möglich, Mockups selbst zu erstellen. Dazu kann der Recorder SceneBuilder[1], ein Programm zur Erstellung graphischer Oberflächen, automatisch öffnen, wodurch der Kunde zusammen mit dem Entwickler Mockups erstellen kann. Diese können dann wieder in den Recorder geladen werden, wodurch sich auch bereits existierende Mockups editieren lassen. Sollte der Kunde lieber auf Papier oder in einem externen Programm seine gewünschte Oberfläche zeichnen wollen, so kann der Recorder auch Bilder importieren, um diese in Mockups umzuwandeln.

Die aufgenommenen Interaktionen lassen sich im Recorder abspielen. Bei der Aufnahme werden nur markante Interaktionen, wie Mausklicks und Tastatureingaben, aufgenommen, um die Länge des Videos kurz zu halten. Auch wird deshalb die Mausbewegung zwischen den Elementen interpoliert. Diese Interaktionen können als Text in Form eines Use-Cases oder in Form eines Videos exportiert werden. Bei der Video Option wird das Fenster des Mockups aufgenommen, während dieses die Interaktion abspielt.

Ein Zustandsdiagramm zum besseren Verständnis des Mockup Recorders ist im Anhang (siehe A.3). Es stellt die Optionen, die der Benutzer bei der Aufnahme von Interaktionen und Erstellung von Mockups hat, sowie das Systemverhalten dar.

[1] `https://gluonhq.com/products/scene-builder/`

Zusammenfassung

	ViViRecorder	**MockupRecorder**
Funktion im RE	Verifikation von Anforderungen	Erhebung, Validierung und Dokumentation
Hauptfunktion	Aufnahme von Videos	Aufnahme von Interaktionen
Struktur	Szenario-Struktur	Szenario-Struktur
Funktion Videoplayer	Abspielen der Videos	Abspielen von Interaktionen
Produkt	Szenario in einem gesamten Video	Szenario als Text, Bild oder Video
Besonderheiten	Unterstützung durch Video-Guidelines	Erstellen und bearbeiten von Mockups
	Umwelt- und Fehleranalyse	Feedback zu graphischen Oberflächen

Tabelle 3.1: Zusammenfassung der Konzepte

3.3 Technische Implementierungen

In der Umsetzung weisen beide Programme viele Gemeinsamkeiten auf. Beide wurden mit der Programmiersprache Java der Version 8 geschrieben, wodurch sie plattformunabhängig sind. Auch werden in beiden Programmen das JavaFX Framework für die graphische Oberfläche verwendet. Dadurch ist in beiden Fällen das *Model-View-Controller* Design Pattern (MVC) vorhanden, wodurch die Logik der Programme von ihrem Design getrennt ist und somit eine leichtere Modifikation möglich ist. So verwenden beide auch die dazugehörigen FXML-Dateien für die graphische Repräsentation.

Zuletzt verwenden beide Programme JavaCV[2], welches eine Java-Schnittstelle zu OpenCV und FFmpeg ist und eine Reihe an Funktionen zur Aufnahme und Bearbeitung von Videos bietet. In beiden Fällen werden diese verwendet, um Videos im MP4-Format zu generieren. Beim ViViRecorder wird dies zur Detektion von Video- und Audioinput sowie zur Aufnahme von Videos mit der Kamera genutzt, beim Mockup Recorder wird mithilfe der Library der Bildschirm aufgenommen.

[2] `https://github.com/bytedeco/javacv`

Im folgenden Punkt werden die unterschiedlichen technischen Implementierungen der Tools aufgezählt:

ViViRecorder

Der ViViRecorder verwendet neben JavaCV auch die jni4net[3] Library, eine Schnittstelle zwischen Java und dem .NET System von Microsoft. Dadurch lässt sich auf die Sensorik des Microsoft Surfaces zugreifen, für welches das Programm entwickelt wurde. Dies hat den Nachteil, dass die Funktionen der Umgebungs- und Fehlerprüfung nur auf dem Surface funktionieren. Zudem startet das Programm nicht, wenn es keine Kameras findet, wodurch optimales Arbeiten nur auf dem Surface garantiert ist. Zum Speichern von Nutzerdaten, wie Namen oder Reihenfolge der Schritte, wird eine SQLite-Datenbank verwendet, welche beim Start des Programms das zuletzt bearbeitete Szenario lädt. Zur Verwaltung aller verwendeten Libraries und Ressourcen wird das Build-Management-Tool Maven[4] verwendet. Es gibt an, wie die Software gebaut wird und welche Abhängigkeiten sie hat. So werden die notwendigen Libraries automatisch heruntergeladen und beim Export in die ausführbare jar Datei kopiert.

Mockup Recorder

Der Mockup Recorder verwendet zum Speichern der Szenarien und Interaktionen die Jackson Library[5]. Diese serialisiert und deserialisiert die Daten in das JSON Format. Dadurch lassen sich die im selbst erstellten .szenario Format gespeicherten Interaktionen leicht sowie platzsparend austauschen und sind sowohl vom Menschen als auch vom Computer lesbar. Anders als der ViViRecorder wird Maven nicht verwendet. Stattdessen wurden die verwendeten Libraries manuell heruntergeladen und importiert. Da dieses Tool keine Kamera oder Sensorik benötigt, funktioniert es auch auf anderen Betriebssystemen, wie zum Beispiel Mac und Linux.

[3] `http://jni4net.com/`
[4] `https://maven.apache.org/`
[5] `https://github.com/FasterXML/jackson`

Zusammenfassung

	ViViRecorder	MockupRecorder
Gemeinsamkeiten	Java 8 JavaFX Framework JavaCV Library MVC Design Pattern	
Unterschiede	jni4net Library SQLite DB Maven	Jackson Library .szenario Format plattformunabhängig

Tabelle 3.2: Technische Gemeinsamkeiten und Unterschiede

3.4 Schnittstellen

In diesem Abschnitt beziehen sich Schnittstellen sowohl auf Eingabe- und Ausgabeschnittstellen sowie auf angrenzenden Systeme.

Vision Video Recorder

Wie bereits erwähnt sind Videos das Eingabe- und Ausgabeformat des Tools. Diese sind im MP4-Format, sodass sie auch in anderen Videoplayern abspielbar sind. Dadurch lassen sich in der Vergangenheit aufgenommene oder extern erstellte Videos leicht importieren und hinzufügen. Die aufgenommenen Videos sowie das beim Export erzeugte Gesamtvideo werden im Ordner des dazugehörigen Szenarios gespeichert.

Da die Szenarien in einer SQLite-Datenbank gespeichert werden, wird neben den Videos auch die Datenbank benötigt, um das Projekt an einem anderen Gerät zu öffnen.

Bei Erkennung von Fehlern während der Aufnahme werden Log-Dateien im TXT-Format im gleichen Ordner des Videos erstellt. In diesen ist der Zeitpunk des Fehlers sowie der Fehlergrund angegeben. Die Logs enthalten den gleichen Namen wie der dazugehörige Schritt, um diese leicht zuordnen zu können.

Mockup Recorder

Der Mockup Recorder erlaubt den Import einerseits von FXML-Dateien, welches das von JavaFX genutzte Format für die graphischen Benutzeroberflächen ist, um die Mockups in die Mockup Pane zu laden.

Dadurch lassen sich die Mockups testen und Interaktionen mit ihnen aufnehmen. Andererseits können auch Bilder im PNG- oder JPG-Format verwendet werden, falls die Benutzeroberfläche auf Papier oder einem Prototyping Programm wie Pencil[6] entwickelt wurde. Der Mockup Recorder erstellt aus diesen Bildern automatisch ein Mockup, in welches die Bedienelemente nur noch manuell hinzugefügt werden müssen, um Interaktionen zu erlauben. Als Ausgabeformate gibt es den schriftlichen Export als Use-Case, in welche die Aktionen des Nutzers und die Antworten des Systems angegeben werden, den Export des Mockups als Bild im PNG-Format sowie die Aufzeichnung der Interaktionen im MP4-Format.

Um die Mockups zu erstellen beziehungsweise bearbeiten, erlaubt der Mockup Recorder das Öffnen dieser in SceneBuilder. Nach einmaliger Angabe des Pfades zum SceneBuilder wird dieser per Klick automatisch gestartet, wodurch sich ohne großen Zeitaufwand Prototypen entwickeln und bearbeiten lassen. Die Änderungen müssen jedoch manuell mit der im Recorder vorhandenen Option neu geladen werden.

Die Szenarien sowie deren enthaltenen Interaktionen werden in einem dafür entwickelten Szenario Modell im JSON-Format gespeichert. Diese enthalten den Pfad zum jeweiligen Mockup, die Reihenfolge im Szenario sowie die Bedienelemente mit ihren Interaktionen, wie Texteingabe oder Mausklicks. Dadurch lassen sich diese leicht mit anderen Personen teilen.

Zusammenfassung

	ViViRecorder	**Mockup Recorder**
Eingabe	MP4	PNG /JPG, FXML
Ausgabe	MP4	MP4, PNG, Use-Case
Medium zum Speichern	SQLite-DB	.szenario Datei im JSON-Format
Zusatz	Logging von Fehlern in einer TXT-Datei	Erstellen und Editieren von FXML-Dateien mit SceneBuilder

Tabelle 3.3: Zusammenfassung der Schnittstellen

[6] `https://pencil.evolus.vn/`

3.5 Vergleich

In diesem Abschnitt wird erläutert, welche Gemeinsamkeiten die Programme haben, in denen sie sich automatisch überlagern können, und in welchen Punkten ein zusätzlicher Aufwand zur Vereinigung aufgebracht werden muss.

Beide Tools werden für das Requirements Engineering verwendet und helfen bei der Erhebung beziehungsweise Validierung von Anforderungen. Dazu benutzten sie Videos, die mit dem Kunden geteilt werden können. Als Struktur verwenden beide dafür Szenarien, in denen Schritte definiert und Handlungen aufgenommen werden. Es existieren bei beiden Tools Funktionen, die das Erstellen, Bearbeiten und Löschen von Schritten erlauben. Diese werden in einer Liste angezeigt, welche in beiden Fällen eine chronologischen Timeline entsprechen. Auch ist die Bedienung innerhalb dieser Timeline gleich, in beiden Tools erfolgt die Bearbeitung per Drag & Drop. Funktionen, die der Mockup Recorder zur Navigation in der Timeline hat, lassen sich auf die Timeline des ViViRecorders anwenden. Zudem lässt sich der Ablauf zur Erstellung der Videos kombinieren, da der Benutzungsablauf in beiden Programmen ähnlich ist. Dabei ist es wünschenswert, die per Kamera aufgenommenen Videos mit den Videos der Mockup Interaktionen zu kombinieren, um verschiedene Ansichten zu zeigen.

In beiden Fällen ist die Wiedergabe von Videos in einem Videoplayer vorhanden, jedoch erlaubt der Mockup Recorder nur die Wiedergabe der Interaktionen. Für die Wiedergabe werden aber ähnliche Funktionen verwendet, wie Start/Stopp, Aufnahme oder Pause. Die Videowiedergabe passt dabei in beiden Fällen in das gleiche Fenster.

Da beide Programme sowohl Java der gleichen Version, als auch JavaFX verwendet haben, ist eine Kombination beider Programme möglich. Die Trennung der graphischen Oberfläche von der Anwendungslogik durch das MVC-Pattern des JavaFX Frameworks erleichtert dabei diese Kombination. Somit lassen sich auch graphischen Oberflächen, wie das die Hauptfenster und Fehlerdialoge, wiederverwenden. Die Verwendung von JavaCV zum bearbeiten und erstellen von Videos in beiden Fällen reduziert den Aufwand ebenfalls, da somit JavaCV weiterverwendet werden kann.

Es bestehen aber auch Konflikte, die aufgelöst werden müssen. Zum einem gibt es Unterschiede in der Szenario-Struktur und dessen Modellierung. Die Modelle benötigen eine gemeinsame Oberklasse, um das Hinzufügen sowohl von Videos als auch von Mockups in dieselbe Timeline zu erlauben. Zudem muss das zentrale Fenster, in dem die Videos und Mockups angezeigt werden, angepasst werden, um die Wiedergabe in separaten Fenstern zu erlauben. Dies führt zu Änderungen der Anwendungslogik, die dafür angepasst werden muss, wobei die alten Funktionen erhalten bleiben sollen.

Ein weiterer Konflikt besteht in der Art der Speicherung von Szenarien.

Die Art und Weise, wie der Mockup Recorder die Interaktionen speichert, ist nicht kompatibel mit der SQLite-Datenbank, da in diesen die Listen der Interaktionen nicht problemlos gespeichert werden können. Dies führt zu einem höheren Aufwand, da die Serialisierung in das JSON-Format sowie die Deserialisierung dieser umgeändert werden müsste. Es muss also entweder eine der beiden Arten angepasst oder eine neue Art der Speicherung gefunden werden, die Kompatible mit beiden Methoden ist.

Dies alles gilt es in einer gemeinsamen Oberfläche zu verbinden, sodass nur ein Tool zur Produktion von Videos benötigt wird und erlernt werden muss.

Gemeinsamkeiten und Konflikte der Tools		
Gemeinsamkeiten	Java & JavaFX	Gleiche Bedienung der Timeline-Elemente
	Unterstützen beim RE	Ausgabe eines Videos
	Szenario-Struktur	Ähnliche Timeline-Navigation
	Ähnlicher Ablauf	Ansicht in einem zentralen Fenster
	Wiedergabe von Videos	Gleiche Video-Libraries (JavaCV)
Konflikte	Unterschiede in der Modellierung der Szenarien	Änderungen der Anwendungslogik
	Anpassung des zentralen Fensters	Art der Speicherung
	Zusammenschneiden von Mockups und Videos	Kombination aller Funktionen mit guter Usability

Tabelle 3.4: Vergleich beider Tools

Kapitel 4

Gesamtkonzept des Rahmenwerks

In diesem Kapitel wird das Gesamtkonzept der zu entwickelnden Software beschrieben. Dabei wird entschieden, wie sich die beiden in den vorherigen Kapiteln analysierten Programme bestmöglich kombinieren und erweitern lassen, um das in Kapitel 1.2 genannte Ziel zu erreichen.

4.1 Umgebung

Das Tool zum erstellen von Videos soll für das Microsoft Surface entwickelt werden, wodurch arbeiten unabhängig vom Standort möglich ist. Das Surface benutzt das Windows 10 Betriebssystem und erlaubt neben der Nutzung mit Tastatur als Laptop auch die Nutzung per Touch im sogenannten Tablet-Modus. Neben einer Kamera auf der Vorder-und Rückseite hat es auch Sensoren, die die Lichtverhältnisse, Neigung und Umgebungslautstärke aufnehmen können. Zur Bedienung können sowohl Tastatur und Maus als auch Touch-Input entweder mit den Fingern oder auch mit dem extra vorhandenen Surface Stift verwendet werden. Der Touch-Input unterscheidet dabei zwischen Fingern und dem Stift, welcher als Mausklick registriert wird.

Das Surface der vierten Generation, welches für die Entwicklung benutzt wird, hat ein 12.3 Zoll großen Bildschirm und eine Auflösung von 2.736 x 1.824 Pixel. Sofern nicht deaktiviert, wechselt der Bildschirm vom Landscape-Mode zum Portrait-Mode und umgekehrt basierend auf der Bildschirmausrichtung. Falls keine Tastatur vorhanden ist, kann das Soft-Keyboard des Surfaces zur Eingabe verwendet werden.

Das Verhalten von der Software unterscheidet sich im Laptop- und Tablet-Modus. Während Programme sich im Laptop-Modus standardmäßig wie auf PCs verhalten, wird im Tablet-Modus je nur eine Ansicht angezeigt, dafür immer im Vollbildmodus, ähnlich wie auf mobilen Geräten. Dialoge werden dabei im Tablet-Modus ebenfalls im Vollbildmodus angezeigt.

4.2 Konzeptentscheidungen

4.2.1 Grundlegendes

Die Grundidee ist, Videos zur Repräsentation von Szenarien zu erstellen, die für die Elicitation und Documentation benutzt werden können. So soll es einfach möglich sein, dem Kunden die Videos für Feedback zu zeigen oder den Entwicklern eine Referenzbasis zu bieten. Diese Szenarien sollen mit Videos beschrieben werden, wobei entweder die reale Welt oder der Bildschirm aufgenommen werden soll. Diese Szenarien beschreiben einen linearen Ablauf. Die Schritte des Szenarios sollen in einer Timeline chronologisch angeordnet sein und am Ende des Prozesses in ein Gesamtvideo exportiert werden. Ein Item in der Timeline entspricht dabei einem Schritt eines Szenarios.

Das Erstellen von Vision Videos soll relativ einfach und ohne großen Aufwand möglich sein. Dazu sollten alle Funktionen in einem zentralen Fenster vorhanden sein, in welchem alle Informationen, wie das gerade ausgewählte Item oder die bereits vorhandenen Items, angezeigt werden. Damit der Requirements Engineer auch zusammen mit dem Kunden arbeiten kann, soll die Benutzeroberfläche einfach verständlich und selbsterklärend sein. Aus diesem Grund soll, ähnlich wie bei mobilen Geräten, vermehrt Icons statt Text verwendet werden. Das Design soll somit an mobile Geräte orientiert sein und wenn passend das Material Design[1] verwenden, welches eine von Google entwickelte Designsprache ist. Somit können Benutzer ihr Vorwissen von Android Geräten auf das Programm anwenden, wodurch ihnen Vertrautheit geboten wird. Zusätzlich wird somit die Verwendung im Tablet-Modus vereinfacht, da die Bedienung mit Icons als Knöpfe ebenfalls aus mobilen Geräten bekannt ist.

4.2.2 Hauptfunktionen

Ziel ist es, die im letzten Kapitel 3 genannten Hauptfunktionen des ViVi-Recorders und Mockup Recorders zu vereinigen. So soll es möglich sein, in einem Programm zwischen der Aufnahme von Videos mit der Kamera und der Aufnahme von Interaktionen mit den Mockups zu wählen und beide Formen in einem Gesamtvideo zu exportieren. Der Wechsel zwischen den Optionen soll dabei mühelos und schnell funktionieren.

Die Abbildung 4.1 soll das im folgenden Verlauf beschriebene Konzept visualisieren. Auf Druck eines Buttons des *Item Choosers* soll zwischen den verschiedenen Optionen gewählt werden können. So werden bei der Wahl eines Elements (Video, Mockup) die Anwendungsfunktionen und -optionen in die *Item Pane* geladen und der Item Chooser durch einen Chooser

[1] `https://material.io`

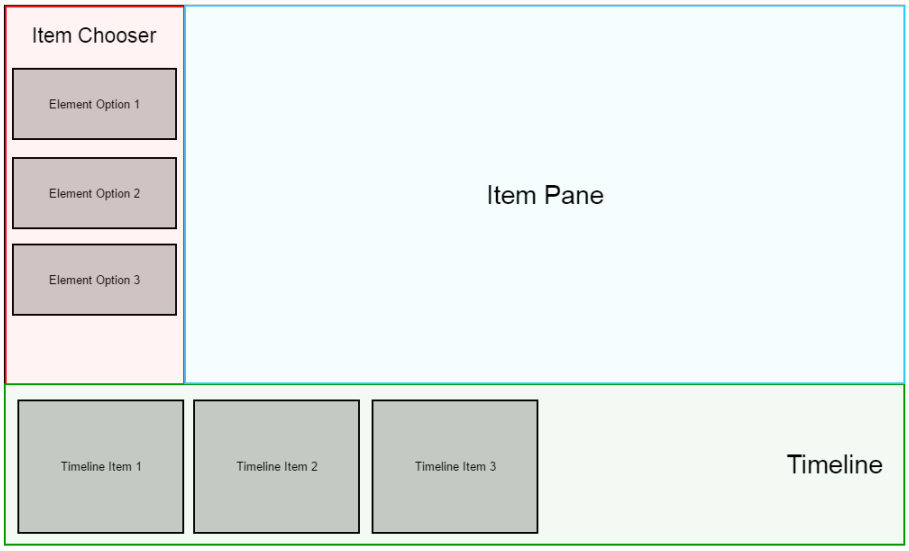

Abbildung 4.1: Veranschaulichung des Konzepts

des jeweiligen Elements ersetzt. Für die Video Option sollen so bereits vorhandene Videos in einer Liste geladen und für Mockups, ähnlich wie in Abbildung 3.4 zu sehen, die bereits erstellten Mockups geladen werden. In der *Timeline* sollen alle definierten Schritte mit einer Vorschau gezeigt werden, falls bereits ein Element zugeordnet wurde. Andernfalls soll durch einen Text gekennzeichnet werden, dass noch kein Element zugeordnet wurde.

Die Bedienung soll, ähnlich wie in den anderen Programmen, per Mausklicks und Drag & Drop möglich sein. So sollen Items in der Timeline problemlos ihre Position und Beschreibung ändern können oder gelöscht werden. Es soll ebenfalls möglich sein, per Knopfdruck neue Elemente zu erstellen und bereits vorhandene zu importieren. Per Rechtsklick auf das jeweilige Timeline Item soll ein Menü mit Funktionen zum Bearbeiten des Items geöffnet werden. Außerdem soll die Option vorhanden sein, die Videos und Mockups in der Item Pane abzuspielen.

4.3 Anforderungen

Für den ViViRecorder und den Mockup Recorder wurden bereits Anforderungen erhoben und die Tools entsprechend diesen entwickelt. Diese Anforderungen sollen im Gesamtkonzept weiterhin erfüllt sein. In diesem Abschnitt werden zusätzliche Anforderungen an das System ermittelt sowie diejenigen, die sich von den original erhobenen Anforderungen unterscheiden oder besonders wichtig sind.

4.3.1 Software

Die zentrale Anforderung an das Gesamtkonzept ist die Option, sowohl Videos wie der ViViRecorder als auch Interaktionen wie der Mockup Recorder aufzunehmen.

[R1] Das System muss die Hauptfunktionen beider Programme enthalten und die Kombination von Videoaufnahmen und Mockupaufnahmen in ein Gesamtvideo erlauben.

Es soll ein Wechsel zwischen diesen Hauptfunktionen im gleichen Fenster möglich sein, damit der Benutzer auf der gleichen Oberfläche arbeiten kann und schnell zwischen Mockups und Videoaufnahme wählen kann.

[R2] Das System muss die Möglichkeit bieten, per Knopfdruck zwischen den verschiedenen Funktionen und Ansichten zu wechseln.

Das System ist für das Microsoft Surface Tablet zu entwickeln und optimieren. Es soll aber auch weiterhin möglich sein, das Programm auf anderen Systemen und PCs zu starten. In diesem Fall soll aber die Videoaufnahme nicht möglich sein.

[R3] Das System muss speziell für den Microsoft Surface entwickelt werden. Es soll mit anderen Windows Geräten kompatible sein.

[R3.1] Das System sollte eine Fehlermeldung beim Aufrufen der Videoaufnahme zeigen, falls es auf einem anderen Windows Gerät oder einem Gerät ohne Kamera läuft.

Ähnlich wie im ViViRecorder sollen alle Szenarien-Schritte vor der Aufnahme definierbar sein. Wurde für einen Schritt noch kein Video aufgenommen oder kein Element zugeordnet, so soll dies in der Vorschau gezeigt werden.

[R4] Das System muss nach dem zuordnen eines Elements zu einem Schritt ein Vorschaubild anzeigen, sonst soll das Fehlen eines Elements durch einen Text kenntlich gemacht werden.

Die Schritte sollen in der gleichen Timeline vorhanden sein und es soll möglich sein, frei zwischen den Items wählen zu können.

[R5] Das System muss alle Elemente in der gleichen Timeline enthalten und fähig sein, diese unabhängig voneinander auszuwählen.

Das Klicken auf den Items öffnet das Ausgewählte in der Item Pane zusammen mit der dazugehörigen Toolbar.

[R6] Falls der Benutzer einen Doppelklick ausführt, muss das System das gewählte Item in der Item Pane zusammen mit der dazugehörigen Toolbar öffnen.

4.3.2 Video

Das Gesamtkonzept soll Videos für das Requirements Engineering erstellen. Um möglichst gute Videos dafür herzustellen, müssen Anforderungen an diese erfüllt sein. Dabei basieren die Anforderungen auf den von Happe [12] ermittelten Guidelines zur Videoproduktion.

Sowohl aufgenommene Videos als auch Mockups sollen in einer festen Größe aufgenommen werden, um ein Video in einer einheitlichen Größe zu erstellen. Dafür muss die in der Abbildung 4.1 gezeigte Item Pane eine feste Größe haben. Nach Karras [15] sollte die Qualität von Videos mindestens 720 x 480 Pixel sein und maximal 1920 x 1080 Pixel. Durch die Limitationen des Surfaces (Größe, Leistung) wird ein Mittelwert gewählt.

[V1] Die Bildqualität des Videos soll 1024 x 600 Pixel sein.

Weiterhin sollte das Gesamtvideo nicht zu lang sein, um nicht die Aufmerksamkeit der Betrachter zu verlieren. Zudem bewirkt eine kürzere Zeit, dass man sich auf die wesentlichen Details fokussieren muss und schon vorher Gedanken zum Inhalt machen muss [7].

[V2] Die Dauer eines Videos sollte maximal 6 Minuten sein.

Für die bestmögliche Qualität des Videos sollen die Guidelines von Happe angewandt werden. Dazu gehören die Kategorien Bild (Beleuchtung), Perspektive/Bewegung (Ruhige Kamera, Dutch Tilt), Audio (Umgebungsgeräusche, Externes Mikrofon) und Dauer (Gesamtlänge, Puffer, Schnitte).

[V3] Die Video Guidelines sollten bei der Aufnahme automatisch angewendet werden.

Um die Videos leicht zu unterscheiden, soll es dem Benutzer möglich sein, ihnen Namen zu geben. Um sie zusätzlich bei mehreren Aufnahmen einzigartig zu halten, soll der Zeitpunkt der Erstellung an das Ende des Namens ergänzt werden.

[V4] Die Namen der Videos müssen aus der gegebenen Beschreibung des Schrittes und der Zeit der Aufnahme bestehen.

Zudem soll zur einfacheren Unterscheidung der Schritte im Gesamtvideo die Beschreibung als Text in der linken unteren Ecke des Videos vorhanden sein. Dadurch kann der Kunde bei Unklarheiten oder Fehlern einfach auf den jeweiligen Schritt im Szenario zeigen.

[V5] Die einzelnen Videoabschnitte im Gesamtvideo sollte die Beschreibung als Text in der linken unteren Ecke enthalten.

4.4 Erweiterungen

Neben den bereits genannten Programmen soll auch das Zeichenprogramm **DrawerTools** integriert werden. Der Quelltext wurde vom Betreuer der Arbeit bereitgestellt. Das in Java geschriebene Programm erlaubt das freie Zeichnen per Mausklick sowie das Hinzufügen von Formen wie Quadraten und Kreisen in die Leinwand. Auch lassen sich die gezeichneten Objekte frei bewegen und löschen. Zudem lässt sich die Farbe und Breite der Striche ändern.

Mithilfe der Integration der Logik der DrawerTools soll es möglich sein, auch Skizzen in die Timeline hinzuzufügen oder diese für das Erstellen von Mockups zu benutzen. In Videoform sollen diese standardmäßig, sofern nicht vom Benutzer geändert, für fünf Sekunden zu sehen sein. Die erstellten Bilder sollen auch als PNG-Datei exportierbar sein. Die Größe ist durch die Anforderungen auf 1024 x 600 Pixel beschränkt.

Weiterhin sollen die in den anderen Arbeiten genannten möglichen Erweiterungen implementiert werden, die aus Zeitmangel gestrichen wurden. So wurde ein akustisches Signal beim Start und Stopp der Aufnahme von Happe vorgeschlagen, da die Koordination mit den Schauspielern eine Schwierigkeit darstellte. Auch sollte ein Puffer von fünf bis zehn Sekunden vor und nach jeder Szene aufgenommen werden, um alle wichtigen Informationen zu erfassen. Für den Puffer wird jedoch auch die Option zum Schneiden von Videos benötigt. Zuletzt wurde vorgeschlagen, den Zeitpunkt der Verstöße gegen die Guidelines im Videoplayer zu markieren, um diese leichter zu finden.

Für die Mockups und Skizzen soll die Möglichkeit bestehen, Audio-Kommentare zu ergänzen, da diese im Normalfall kein Audio haben, was vom Nutzer als störend empfunden werden kann. So können wichtige Hinweise oder zusätzliche Kommentare hinzugefügt werden. Diese Kommentare dürfen dabei maximal so lang sein wie die Videos, um nicht den Spielfluss zu stören.

Für den Fall, das der Nutzer einen Schritt in der Timeline hinzufügen will, für den er noch kein Dokument hat oder wofür er noch mehr Informationen benötigt, so soll dies durch eine leere Zeichenfläche in der Timeline dargestellt werden. Sollte dieser Schritt auch während des Exports noch undefiniert sein, so soll dies durch ein Zeichen im Export veranschaulicht werden.

Somit ergeben sich die folgenden weiteren Anforderungen:

[ReqEx 1]	Das System muss dem Nutzer die Möglichkeit bieten, Audio Kommentare für Mockups und Skizzen hinzuzufügen.
[ReqEx 2]	Die maximale Länge der Audio Kommentare darf maximal die Länge der zugehörigen Skizze im Video oder der Länge der Mockup-Interaktionen sein.
[ReqEx 3]	Das System muss die Möglichkeit bieten, Skizzen in die Timeline hinzuzufügen.
[ReqEx 4]	Das System muss fähig sein, die erstellten Skizzen im PNG-Format zu speichern.
[ReqEx 5]	Das System sollte die Möglichkeit bieten, die erstellten Skizzen als Mockups im SceneBuilder zu editieren.
[ReqEx 6]	Skizzen sollen als Video exportiert standardmäßig fünf Sekunden lang sein.
[ReqEx 7]	Das System muss die Möglichkeit bieten, die Länge der Skizzen im Video zu ändern.
[ReqEx 8]	Das System muss fähig sein, Skizzen in der Item Pane zu zeichnen.
[ReqEx 9]	Das System muss fähig sein, bereits existierende Bilder zu importieren.
[ReqEx 10]	Das System sollte die Möglichkeit bieten, Videos im Programm zu schneiden.
[ReqEx 11]	Das System muss die Möglichkeit bieten, undefinierte Schritte in die Timeline hinzuzufügen. Das System muss diese im Export durch einen Text veranschaulichen.

Tabelle 4.1: Zusammenfassung der Anforderungen der Erweiterungen

Das Zustandsdiagramm zum Gesamtkonzept ist im Anhang.

4.5 Anwendungsfall

Der Hauptanwendungsfall ist die Erstellung von Videos zur Visualisierung von Szenarien. Zu einer besseren Vorstellung der Arbeitsweise soll das Werkzeug in einem konkreten Szenario dargestellt werden. In diesem Fall will der Kunde ein neues System zum Kaufen von Fahrkarten entwickeln, welches erlaubt, per Kontakt mit dem Smartphone zu bezahlen. Das Gesamtkonzept findet Anwendung nach der Erhebung der Anforderungen mit dem Kunden und nach erster Interpretationen durch die Anforderungsingenieure. Die Requirements Engineers haben für die Anforderungen bereits Szenarien schriftlich entwickelt, erste Mockups sowie Skizzen erstellt und Videos, die sie bereits besitzen und wiederverwenden können, gesammelt. Die Visualisierung geschieht dabei im Büro des Anforderungsingenieurs, er kann aber auch durch die Mobilität des Surfaces mit dem Kunden zusammen oder von unterwegs aus arbeiten.

Nach dem Start des Gesamtkonzepts und dem Erstellen des neues Szenarios „Fahrkarte kaufen“ werden die bereits erstellten Dokumente in die jeweiligen Ordner manuell kopiert und das Programm neu geladen.

Für den ersten Schritt will der Engineer eine Person zeigen, die zum Fahrkartenautomat geht. Da er dafür noch kein Video hat, klickt er auf den „Neues Video erstellen“ Button. Daraufhin wird die Kamera geöffnet und er kann zusammen mit einem Schauspieler diese Handlung aufnehmen. Während der Aufnahme wird die Umgebung überprüft und der Benutzer bei Verstößen gegen die Guidelines am Ende der Aufnahme informiert. Da der Benutzer nach Aufnahmeende die Fehlermeldung „Zu dunkel“ bekommt, klickt er auf das Symbol zum Öffnen des Videoplayers, um das Video manuell zu überprüfen. Er stimmt dem Fehler zu und löscht per Druck auf den Löschknopf das Video und geht zurück zur Kamera-Ansicht, um das Video neu aufzunehmen.

Nachdem der Benutzer fertig ist mit der Aufnahme und die Kamera geschlossen hat, öffnet sich ein Dialog, in welchem die Beschreibung des Schrittes bestimmt wird. Der Benutzer tippt „Nutzer geht zum Automaten“ und schließt den Dialog. Daraufhin wird der Schritt in die Timeline mit dem gegebenen Namen und Video hinzugefügt.

Als nächsten Schritt will der Engineer die graphische Oberfläche des Startbildschirms zeigen. Dafür eignet sich ein Mockup am besten. Er zieht das dafür produzierte Mockup in die Timeline und gibt dem Schritt den Namen „User wählt Fahrkarte kaufen“. Danach nimmt der Benutzer die Interaktion per Druck auf den Aufnahme-Knopf auf, in diesem Fall den Klick auf den Menüeintrag „Fahrkarte kaufen“.

Für die Interaktion zur Auswahl des Tickets wird ein anderes Mockup aus dem Chooser in die Timeline gezogen. Dies erhält die Beschreibung „Nutzer wählt Zone und Datum“. Im darauffolgenden Schritt will der Benutzer die Auswahl des Bezahlvorgangs mit dem Smartphone veranschaulichen. Dafür zieht er das dazugehörige Mockup in die Timeline, gibt dem Schritt die Beschreibung „Nutzer wählt bezahlen mit Smartphone“ und nimmt den Klick als Interaktion auf.

Im nächsten Schritt soll der Bildschirm mit dem Preis und ein Bild vom Bereich angezeigt werden, an welches das Smartphone gehalten werden soll. Dafür öffnet der Benutzer den Chooser mit den Bildern und fügt das gewählte Bild in die Timeline hinzu. Als Beschreibung wählt er „System zeigt Preis und Scanner Bereich“.

Da der Engineer aus den Anforderung nicht erschließen kann, welche Zahlungssysteme, wie Apple[2] oder Google Pay[3], unterstützt werden sollen, will er in einem Audiokommentar dies an dieser Stelle anmerken. Dafür öffnet der Benutzer per Rechtsklick das Menü des Timeline Schrittes und wählt „Audiokommentar hinzufügen“. Es öffnet sich ein Dialog mit der Aufnahmeansicht. Er klickt auf den Aufnahme-Knopf und spricht „Hier benötigen wir mehr Informationen zu den Zahlungssystemen, die Sie unterstützen wollen“. Danach speichert er diese Audio-Datei per Knopfdruck und schließt das Dialog.

Daraufhin soll veranschaulicht werden, wie der Benutzer das Smartphone an den Automaten hält. Das Video dazu wurde bereits zuvor mit dem anderen Video aufgenommen. Also öffnet er den Video Chooser und zieht das Video in die Timeline und gibt dem Schritt die Beschreibung „User bezahlt das Ticket“.

Da dem Engineer ebenfalls aus den Anforderungen unklar ist, was nach dem Bezahlvorgang passieren oder wie es aussehen soll, entscheidet er sich, dies im Video zu veranschaulichen. Dafür fügt er ein undefinierten Schritt mit der Beschreibung „System zeigt Erfolgsmeldung und druckt Ticket“, welches im Export das Fehlen von Informationen veranschaulicht.

Nachdem alle Schritte des Szenarios hinzugefügt wurden, spielt der Benutzer die Timeline per Klick auf den Menüeintrag „Timeline abspielen“ ab, um vor dem Export das erstellte Szenario zu überprüfen. Da keine Probleme vorhanden sind, klickt der Benutzer auf „Exportieren“. Nach dem Erstellen der Teilvideos und dem Zusammenschneiden wird der Dialog zum erfolgreichen Export angezeigt. Per Knopfdruck auf „Zeige Datei“ kommt der Benutzer zum Speicherort des Gesamtvideos, der generierten Teilvideos und der .szenario-Datei. Das Gesamtvideo kann er zum Abschluss dem Kunden zur Verifikation senden. Daraufhin können neue Szenarien visualisiert werden.

4.6 Verwandte Arbeiten

Neben den in dieser Arbeit implementierten Arbeiten haben sich schon andere Forscher mit der Thematik zur Erstellung von Szenarios in Videoformat im Requirements Engineering beschäftigt. In diesem Abschnitt wird auf einige verwandte Arbeiten eingegangen und wie sich diese Arbeit von denen unterscheidet.

[2] `https://www.apple.com/de/apple-pay/`
[3] `https://pay.google.com/intl/de_de/about/`

Creighton et al. [7] haben sich in *Software Cinema—Video-based Requirements Engineering* mit der gleichnamigen Technik zur Erstellung von Videos zur Beschreibung von Szenarien befasst. Diese werden genutzt, um den Stakeholdern das Gesamtsystem visuell zu präsentieren und den Entwicklern eine Referenzbasis zu bieten. Dafür wurde das *Software Cinema tool kit* entwickelt, ein Editor, mit welchem Annotationen hinzugefügt und UML-Diagramme erstellt werden können. Im Unterschied dazu befasst sich diese Arbeit ebenfalls mit der Erstellung von Videos, jedoch soll der Prozess der Generierung sowohl für den Entwickler als auch für den Stakeholder vereinfacht und einfach verständlich sein.

Kitzmann [20] hat in *Konzept und Implementierung eines Werkzeugs für multimediale Anforderungserhebung und -validierung* den *Vision Catcher* entwickelt, mit welchem während des Kundeninterviews Feedback des Kunden direkt umgesetzt werden kann. So soll die Einarbeitung des Kunden in die Dokumentationsform verkürzt und die Digitalisierung von Dokumenten vereinfacht werden. Die Verständlichkeit soll so erhöht und der Kunde zu weiteren Anforderungen angeregt werden.

Ebenfalls wie in dieser Arbeit werden dafür multimediale Dateien verwendet. Anders all der Vision Catcher erlaubt das Gesamtkonzept dieser Arbeit die Dokumentation in Videoform und die Interaktion mit zuvor erstellen Mockups.

Stangl [29] hat sich in *Script: A Framework for Scenario-Driven Prototyping* mit dem von ihm erstellten *SCRIPT* Editor zur Verwendung von Szenarien im Prototyping beschäftigt. Ähnlich wie der Mockup Recorder [11] lassen sich mit dem Editor interaktive Prototypen aus Grafiken erstellen. Außerdem lassen sich Szenarien erstellen und das Endergebnis in einem Video exportieren. Im Unterschied zum SCRIPT Editor ist der Fokus dieser Arbeit die Erstellung von Vision Videos mit möglichst wenig Aufwand durch den Entwickler und Stakeholder, sodass schnell Feedback gegeben werden kann.

Ebenfalls Stangl und Creighton [30] haben in *Continuous Demonstration* einen Ansatz zur Veranschaulichung des vorhandenen Systems im gesamten Entwicklungszeitraum entwickelt. So sollen Videos, die mit möglichst geringem Aufwand herzustellen sind, dabei helfen, Feedback von den Stakeholdern zu bekommen. Dies soll es erlauben, auch bei noch nicht vorhandenen Systemen die Konzepte dem Stakeholder zu präsentieren, um möglichst früh Probleme zu finden.

Diese Arbeit baut auf die in *Continuous Demonstration* genannten Anforderungen zur Erstellungen der Videos auf. So sollen die mit dem Gesamtkonzept erstellten Videos die statischen und dynamischen Modelle des Systems veranschaulichen, dem Benutzer soll es ohne großen Aufwand möglich sein, Videos zu erstellen und es soll einfach sein, die vorhandenen Szenarien mit geringem Aufwand anzupassen.

Karras et al. [19] haben sich in *Video as a By-Product of Digital Prototyping: Capturing the Dynamic Aspect of Interaction* mit der Erstellung von Videos als Nebenprodukt des Durchlaufs von interaktiven Mockups beschäftigt. In der darin durchgeführten Evaluation wurde gezeigt, dass die aufgezeichneten Interaktionen im Vergleich zu statischen Veranschaulichung zu einem schnelleren Verständnis beim Benutzer führen. In dieser Arbeit soll auf diese Ansätze aufgebaut und durch die Unterstützung von multimedialen Dokumenten erweitert werden.

Kapitel 5

Implementierung des Gesamtkonzepts

In diesem Kapitel wird auf Basis der im letzten Kapitel 4 ermittelten Anforderungen und Analyse das entwickelte Gesamtkonzept vorgestellt. Da die Hauptaufgabe des Gesamtkonzepts die Erstellung von Vision Videos ist, wird das Programm im Folgenden als **Vision Video Maker** bezeichnet.

5.1 Technische Grundlagen

Der Vision Video Maker wurde in der Programmiersprache Java der Version 8 für das Microsoft Surface entwickelt. Es basiert auf den Funktionen des Mockup Recorders und des ViViRecorders, wobei besonders der Mockup Recorder als Ausgangspunkt für die weitere Entwicklung genutzt wurde. Weiterhin wurde JavaFX für die Erstellung der graphischen Oberflächen verwendet. Zur Verwaltung dieser sowie weiterer Ressourcen wurde das Management Werkzeug Maven angewandt.

Für die Videoaufnahme wurde die in den anderen Arbeiten bereits genutzte JavaCV Library weiter verwendet, wobei auf die zum Stand dieser Arbeit aktuellste Version aktualisiert wurde. Neben den in den anderen Arbeiten genutzten Libraries, wie Jackson zur Speicherung der Daten im JSON-Format oder der jni4net Library für den Zugriff auf die Sensorik des Surfaces, enthält der Vision Video Maker das Kommandozeilenprogramm FFmpeg[1], welches zur Bearbeitung der Videos genutzt wird.

Zusätzlich wurden für graphische Erweiterungen JFoenix[2], eine Material Design Library für JavaFX, sowie ControlsFX[3], eine Library für weitere UI Elemente für JavaFX, verwendet.

[1] `https://ffmpeg.org/`
[2] `https://github.com/jfoenixadmin/JFoenix`
[3] `https://github.com/controlsfx/controlsfx`

Zum Speichern und Laden der Szenarien wurde das .szenario-Modell vom Mockup Recorder erweitert. Es wird weiterhin die Jackson-Library zum Serialiseren und Deserialisieren der Daten im JSON-Format verwendet. Neben den Interaktionen der Mockups lassen sich nun auch Videos und Bilder in der Datei speichern. Zusätzlich enthält das Szenario-Modell auch die Informationen, die der ViViRecorder in dessen Modell speicherte, wie der Beschreibung des Schrittes oder dem Namen des Szenarios. Für jeden Schritt in dem Szenario wird der Pfad zu der Datei, die Länge im Video sowie der Pfad zu der dazugehörigen Audio-Datei gespeichert.

Die Videos werden weiterhin im MP4-Format exportiert. Erstellte Skizzen und Bilder können sowohl im PNG-Format gespeichert werden als auch als Videos exportiert werden. Zusätzlich lassen sie sich durch die Funktion des Mockup Recorders in FXML-Dateien konvertieren. Für die aufgenommenen Audiodateien wird das MP3-Format verwendet, wodurch sie sich auch in externen Audioplayern abspielen lassen.

5.2 Design- und Entwurfsentscheidungen

In diesem Abschnitt werden das Aussehen der Oberfläche und die implementierten Features genauer beschrieben. Die Abbildung 5.1 zeigt das Startfenster und ist die erste Ansicht, die der Benutzer zu sehen bekommt. Von dieser aus kann über das Anwendungsmenü ein neues Szenario erstellt oder ein bereits existierendes geladen werden. Die Hauptansicht wurde vom Mockup Recorder übernommen, wobei der Mockup Chooser durch ein JavaFX Accordion mit Choosern für Videos, Mockups und Bilder ersetzt wurde. Durch die Weiterverwendung der Split Panes lassen sich der Element Chooser (rechte Seite) und die Timeline (untere Hälfte) verschieben, um mehr Platz zu bekommen. Dies ist wegen der limitierten Größe des Surfaces notwendig, da das zentrale Fenster und die Timeline nicht gleichzeitig offen seien können. Die Position der unteren Split Pane lässt sich per Menüeintrag dynamisch ändern. (siehe Abbildung 5.2 für die offene Timeline Ansicht).

Element Chooser

Der Element Chooser ist, wie der Mockup Chooser vom Mockup Recorder, eine Liste aller Dokumente, die im jeweiligen Order des Elements vorhanden sind. Durch den Klick auf eine TitledPane öffnet sich der dazugehörige Chooser. Die Einträge haben ein Vorschaubild sowie den Dateinamen und bieten damit eine einfachen Übersicht. Auch wurde die Doppelklick Funktion, welche das Item in der zentralen Ansicht zusammen mit der dazugehörigen Toolbar öffnet, sowie das Drag & Drop Verhalten übernommen, mit welcher die Dokumente in die Timeline gezogen werden können.

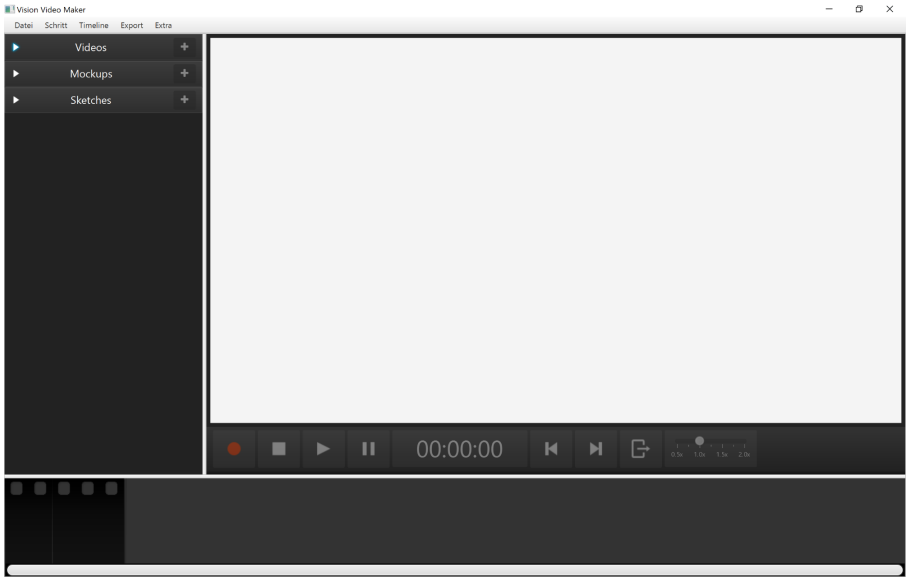

Abbildung 5.1: Startfenster des Vision Video Makers

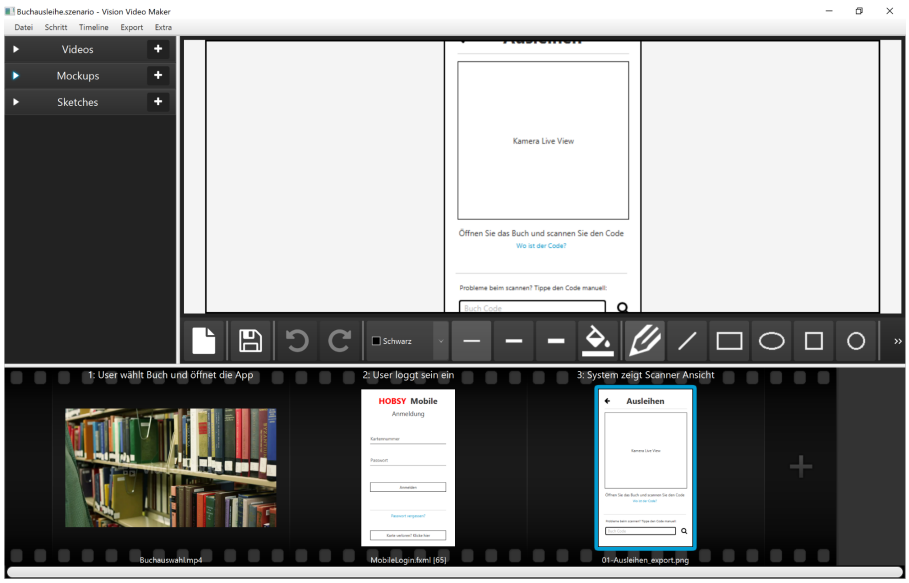

Abbildung 5.2: Ansicht der vollständigen Timeline

Nach dem Drop wird ein Dialog geöffnet, in dem der Benutzer dem neuen Schritt eine Beschreibung geben kann. Per Rechtsklick öffnet sich ein Kontextmenü, mit welchem entweder der Chooser neu geladen oder das ausgewählte Item von der Festplatte gelöscht werden kann.

Der Klick auf den Plus-Button erstellt eine neue Datei des jeweiligen Elements. Im Falle von Videos wird die Kamera geöffnet, für Mockups kann ein Bild ausgewählt werden, welches als FXML konvertiert im SceneBuilder geöffnet wird. Im Falle von Sketches wird eine neue leere Zeichenfläche geladen.

Timeline

Die Timeline, ebenfalls vom Mockup Recorder übernommen, stellt die Schritte des Szenarios chronologisch dar. Neben dem Dateinamen und einem Vorschaubild wurde das Timeline Item mit einem Label für die Beschreibung des Schrittes erweitert. Der Beschreibung wird automatisch die Nummerierung vorangestellt. Weiterhin lassen sich sie Reihenfolge der Schritte durch Drag & Drop ändern und das Item durch Doppelklick im zentralen Fenster mit der dazugehörigen Toolbar öffnen. Zusätzlich haben die Items ein element-spezifisches Kontextmenü, welches durch Rechtsklick auf das Item geöffnet wird. Für alle Items kann man so die Beschreibung ändern oder das Item aus der Timeline entfernen.

Videos haben zusätzlich die Option, das vorhandene Video neu aufzunehmen. Dabei wird die Kamera geöffnet und das alte Video durch die Neuaufnahme in der Timeline ersetzt. Das alte Video wird dabei nicht gelöscht und kann weiterhin von dem Chooser aus verwendet werden.

Bilder haben die Option, die Länge der Ansicht im Video zu ändern. Es wird auf Klick ein Dialog geöffnet, in welchem man die Zeit in Sekunden angeben kann.

Sowohl Bilder als auch Mockups erlauben es per Rechtsklick, ein Audio-Kommentar aufzunehmen. Mehr dazu ist im Abschnitt Audiokommentare.

Element Pane

Die Element Pane ist das zentrale Fenster, in dem das Item geöffnet wird. Beim Öffnen wird die Element Pane vergrößert und die Timeline verkleinert. Je nach Element-Typ wird eine unterschiedliche Pane und Toolbar geladen.

Video Pane

Die Video Pane (Abbildung 5.3) hat als zentrale Ansicht eine MediaView, in welcher das ausgewählte Video abgespielt werden kann. Die Toolbar enthält Buttons zum Togglen eines Loops, zur Wiedergabe des Videos und

einen Button, welcher den Video Editor öffnet, in welchem das ausgewählte Video geschnitten werden kann. Die Buttons haben dafür die typischen Icons, die aus anderen Video-Playern bekannt sind und sollen somit für den Benutzer leicht verständlich sein. Außerdem lässt sich durch Klick auf den Schieberegler zu einem beliebigen Punkt im Video springen und durch Klick auf den Volumen Schieberegler die Lautstärke des Videos ändern. Dies erlaubt es dem Benutzer, Videos im Programm selbst zu betrachten und bearbeiten.

Abbildung 5.3: Ansicht der Video Pane mit dazugehöriger Toolbar

Mockup Pane

Die Mockup Pane (Abbildung 5.4) wurde direkt vom Mockup Recorder übernommen und besitzt die gleichen Funktionen. Es wurden lediglich für dieses Programm unwichtige Funktionen, wie die Änderung des Viewports, entfernt, da die Mockups immer in der gleichen Größe aufgenommen werden.

Sketch Pane

Die Sketch Pane (Abbildung 5.5) ist die Modifikation der DrawingTools. Auf der leeren weißen Leinwand lässt sich frei zeichnen und Formen hinzufügen. Eine Erweiterung ist, dass der Doppelklick auf ein Item das Bild als Hintergrund in die Leinwand lädt, wodurch der Benutzer sich Notizen zum Bild machen oder Änderungen vornehmen kann.

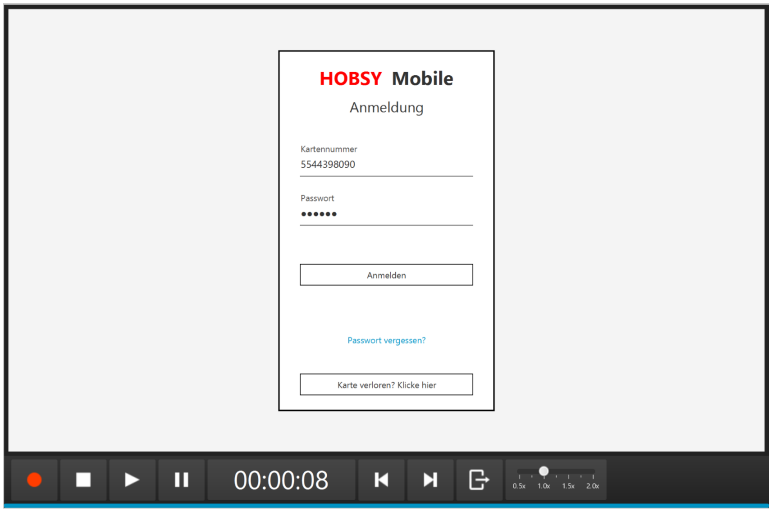

Abbildung 5.4: Ansicht der Mockup Pane mit dazugehöriger Toolbar

Zusätzlich lässt sich das in der Leinwand vorhandene Bild per Knopfdruck im PNG-Format speichern. Ist das ausgewählte Item aus der Timeline, so wird dies durch die neue Version des Bildes ersetzt, wobei die alte Version nicht gelöscht wird und weiterhin im Chooser vorhanden ist. Auch lässt sich das gezeichnete Bild zu einem Mockup konvertieren und anschließend im SceneBuilder öffnen.

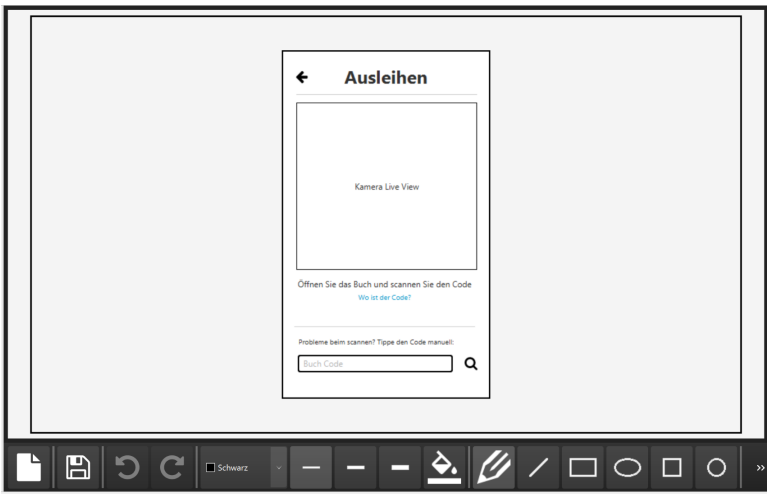

Abbildung 5.5: Ansicht der Sketch Pane mit dazugehöriger Toolbar

Kamera

Die Kamera ist vom ViViRecorder übernommen und in Abbildung 5.6 zu sehen. Es hat die gleichen Funktionen wie das Original, wobei lediglich das Design modifiziert wurde, um zum Rest des Programms zu passen. So sind die Optionen zur Umweltanalyse und zum Wechsel von Video und Audio in einem Drawer versteckt, welcher beim Druck auf den Einstellungs-Button geöffnet wird (siehe Abbildung 5.7). Von dem Drawer aus lassen sich die verschiedenen Inputs ändern und die Umwelt analysieren. Da beide Optionen weniger häufig während der Aufnahme benutzt werden, wurden sie aus der Hauptansicht entfernt, um einen besseren Überblick zu schaffen.

Nach der Aufnahme von mindestens einem Video besteht die Option, die zuvor aufgenommenen Videos in dem Videoplayer der Kamera abzuspielen. Nach dem Schließen der Kamera wird, wenn nur ein Video aufgenommen wurde, ein Dialog geöffnet, wo der Benutzer den Namen der Datei festlegen kann, welcher zeitgleich als Beschreibung des Schrittes dient. Das aufgenommene Video wird danach als neues Timeline-Item hinzugefügt. Wurden mehrere Videos aufgenommen, so wird für jedes der Beschreibungsdialog geöffnet. In diesem Fall werden die Videos aber nur in den Chooser hinzugefügt, damit der Benutzer die passenden Videos auswählen kann.

Abbildung 5.6: Ansicht der Kamera

Videoplayer

In dieser Ansicht (siehe Abbildung 5.8) lassen sich bereits aufgenommene Videos abspielen. Dadurch kann der Benutzer während der Aufnahme-Sitzung die Videos überprüfen und gegebenenfalls per Knopfdruck löschen.

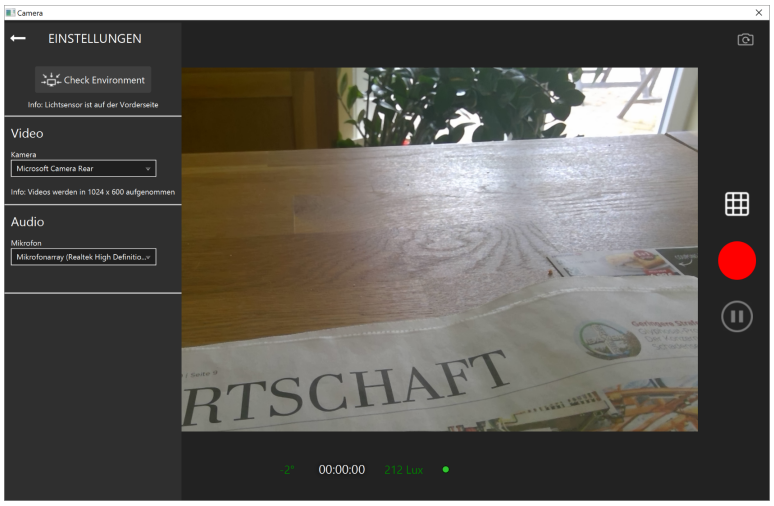

Abbildung 5.7: Ansicht der Kamera mit geöffneten Einstellungen

Auch ist es möglich per Knopfdruck zwischen den Videos zu wechseln. Sollte es Guideline Verletzungen während der Aufnahme gegeben haben, so werden diese mit einem Zeitstempel in der TableView (rechte Seite) angezeigt. Per Doppelklick auf einen Eintrag wird zu der Stelle im Video gesprungen, an der die Verletzung auftrat. Somit kann der Benutzer leicht seine Videos überprüfen und gegebenenfalls per Knopfdruck zurück zur Kamera springen.

Abbildung 5.8: Ansicht des Videoplayers

Editor

Mit dem Editor (siehe Abbildung 5.9) lassen sich Videos schneiden. Dabei ist es möglich, sowohl einen neuen Start- als auch Endpunkt mit dem ControlsFX RangeSlider zu wählen. Zum Testen der neuen Zeiten kann das Video im Editor abgespielt werden, wobei es bei der gewählten Startzeit anfängt und bei der gesetzten Endzeit stoppt. Ist der Benutzer zufrieden mit der Auswahl, so wird auf Klick des Buttons das Video mit FFmpeg geschnitten und in der Video Pane neu geladen. Da das Schneiden permanent ist, wird der Benutzer vorher zur Bestätigung aufgerufen.

Abbildung 5.9: Ansicht des Editors

Audiokommentare

Bilder und Mockups haben kein Audio. Durch den Audioaufnahme-Dialog (siehe Abbildung 5.10) lassen sich für diese Timeline Elemente Kommentare aufnehmen. Die Länge der Aufnahme ist begrenzt durch die Länge, die das Dokument im exportierten Video haben wird. Für Bilder ist das standardmäßig fünf Sekunden, wobei der Benutzer dies durch den dazugehörigen Dialog ändern kann. Wird die Zeit überschritten, so wird die Aufnahme automatisch durch einen Timer gestoppt. Mockups werden während der Aufnahme im Hintergrund abgespielt, wodurch der Nutzer Kommentare zu bestimmten Stellen machen kann. Ist das Abspielen der Interaktionen zu Ende, so wird auch die Audioaufnahme gestoppt.

Durch Klick auf den Aufnahme Knopf wird Audio im MP3-Format aufgenommen. Durch den Schieberegler für Bilder symbolisch als Zeitstrahl beziehungsweise dem Abspielen der Interaktionen im Hintergrund für Mockups kann der Benutzer sehen, wie viel Zeit ihm noch bleibt.

Die aktuellste Aufnahme kann durch Klick auf den Play-Button abgespielt werden, wodurch der Benutzer das Kommentar überprüfen kann. Ist er zufrieden damit, so kann er über den Knopfdruck das Kommentar zum Item hinzufügen. Drückt er auf Abbruch oder schließt das Fenster, so bleibt der alte Zustand erhalten.

Die MP3-Dateien haben zur Unterscheidung sowohl die Beschreibung des zugehörigen Schrittes als auch den Zeitstempel der Aufnahme als Namen. Alte Aufnahmen werden nicht gelöscht, jedoch ist die Auswahl alter Aufnahmen aus Zeitgründen nicht unterstützt.

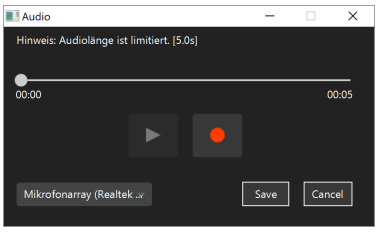

Abbildung 5.10: Ansicht des Audioaufnahme-Dialogs

Export

Ist der Benutzter fertig mit seiner Arbeit, so kann er über den Menüeintrag *Exportieren* die Dokumente der Timeline in ein MP4-Video exportieren. Das Gesamtvideo hat den Namen des Szenarios. Alle im folgenden genannten Schritte werden mit Hilfe von FFmpeg über die Konsole ausgeführt.

Im ersten Schritt werden die Mockups der Timeline als Videos exportiert. Dies geschieht in einem separatem Schritt, um zu verhindern, dass der Export während der Aufnahme zum nächsten Video springt und so die Mockups im Endprodukt fehlen.

Nachdem die Mockups mithilfe der Funktionen des Mockup Recorders aufgenommen wurden, werden sie in die richtige Größe geschnitten und erhalten Stummes Audio[4]. Dies ist notwendig, da sich sonst beim Export die Audiospur der Videos verschiebt.

[4]Audio, welches im Endprodukt nicht hörbar ist. Siehe `https://ffmpeg.org/ffmpeg-filters.html#anullsrc` für mehr Informationen.

Nachdem alle Mockups in Videos umgewandelt wurden, beginnt der eigentliche Export. Für jedes Item in der Timeline wird, basierend auf den Typ des Elements, das generierte Video in die Export-Liste hinzugefügt. Videos und Mockups werden in diesem Schritt nicht bearbeitet, da sie bereits im Videoformat vorhanden sind. Bilder und zu diesem Zeitpunk undefinierte Schritte werden hier in Videos der spezifizierten Länge umgewandelt.

Im Falle von Bildern wird überprüft, ob die Bilder bereits im Format des Videos vorhanden sind. Ist dies nicht der Fall, so wird für diese ein weißer Hintergrund hinzugefügt, sodass das Bild im Endprodukt nicht gestreckt wird. Das Bild ist dabei zentriert.

Im Falle von undefinierten Elementen wird das weiße Bild der Leinwand durch ein Bild mit Text ersetzt, welcher im Endprodukt veranschaulicht, dass für diesen Schritt noch keine Dokumente oder Informationen vorhanden sind. Dies erlaubt es dem Engineer, den Kunden auf fehlende Spezifikationen hinzuweisen.

Im darauffolgenden Schritt wird für jedes Video die Beschreibung als Text in die linke untere Hälfte des Videos hinzugefügt. Dies erlaubt eine bessere Unterscheidung der einzelnen Schritte im Endprodukt. Besitzt das Item ein Audiokommentar, so wird es ebenfalls in das Teilvideo hinzugefügt, andernfalls wird stummes Audio verwendet.

Als letztes werden alle Videos zusammengeschnitten. Der Verlauf der Teilvideoerstellung sowie des Zusammenschneidens wird durch den ControlsFX ProgressDialog veranschaulicht, wodurch der Benutzer Feedback über den Verlauf der Erstellung bekommt. Am Ende des Vorgangs wird der Benutzer durch einen Dialog informiert, mit welchem er zum Speicherort des Videos gehen kann. Zusätzlich befinden sich in einem Ordner im selben Pfad die erstellten Teilvideos, welche für weitere Bearbeitungen benutzt werden können.

5.3 Einschränkungen

Aus zeitlichen Gründen ergaben sich in der Entwicklung Einschränkungen einiger Funktionen, die in späterer Bearbeitung gelöst werden müssen.

Die im Videoplayer vorhandene Liste von Fehlern sollte für die Zeit-Spalte aus Hyperlinks bestehen, sodass der Benutzer zum Klicken auf den Zeitpunk angeleitet wird. Aus Zeitgründen und nicht ganz trivialer Implementierung wurde die gleiche Funktion durch einen Doppelklick ersetzt. Jedoch besteht dadurch die Gefahr, dass der Benutzer diese Funktion nicht bemerkt und benutzt. Es wurde ein Tooltip hinzugefügt, welcher die Doppelklick-Funktion beschreibt, jedoch muss der Benutzer dafür die Maus über die Liste bewegen.

Die Umweltanalyse funktioniert trotz Kopie der Funktionen aus dem ViViRecorder nicht problemlos. Während der Aufnahme werden mit Ausnahme der Audiowerte alle anderen Sensoren überprüft und die Fehler erkannt. Jedoch ist aus unbekannten Problemen die Umweltanalyse per Knopfdruck nicht möglich und aus zeitlichen Gründen war es nicht möglich, den Fehler zu finden. Auch läuft das Programm bei der Nutzung der jni4net Funktionen nach Programmende im Hintergrund weiter.

Aus diesen Gründen sowie um Nutzern von Nicht-Surface Geräten die Nutzung zu erlauben, wird die Sensorik in diesem Prototyp durch eine Option im Menü aktiviert. Der Nutzer wird dabei auf die Risiken der Sensorik-Nutzung hingewiesen. Sollte dieser die Sensorik deaktiviert lassen, so sind Aufnahmen weiterhin möglich, es werden lediglich die Video-Guidelines nicht angewandt.

Eine Liste aller bekannten Fehler und Probleme ist im Anhang.

Kapitel 6

Evaluation

In diesem Kapitel wird die Planung, Durchführung und Auswertung der durchgeführten Evaluation in Form einer Aufgabenstellung betrachtet. Weiterhin wurde die Benutzbarkeit mithilfe eines Fragebogens durch den Probanden bewertet.

6.1 Ziel und Aufbau der Evaluation

Das Ziel der Evaluation ist die Überprüfung der Usability des Vision Video Makers. Dabei wurde untersucht, inwieweit die verwendeten Konzepte und Darstellungen für den Benutzer verständlich waren und wie lange sie benötigten, um ein konkretes Szenario in verschiedenen Teilaufgaben zu visualisieren. Als Metrik wurde die Einarbeitungs- und Bearbeitungszeit der Probanden in Sekunden gewählt. So lässt sich überprüfen, wie weit die Probanden von den geschätzten Aufwandszeiten abweichen, welches Hinweise auf Probleme beim Verständnis durch das Design geben kann.

Zum Erreichen dieses Ziels wurde für die Aufgabenstellung die *Think-Aloud Studie* verwendet. So sollte der Proband während der Bearbeitung laut seine Gedanken äußern, um zum einem Probleme bei der Aufgabenbearbeitung zu finden und zum anderen das mentale Gedankenbild des Probanden zu verstehen. Bei Problemen war es dem Probanden erlaubt, dem Prüfer Fragen zu stellen.

Im zweiten Teil des Evaluationsvorgangs wurde in einem Gespräch mit dem Probanden ein Fragebogen ausgefüllt. In diesem wurden dem Probanden Fragen zu verschiedenen Aspekten der Usability des Programms gestellt, welcher dieser in einer fünf stufigen Likert-Skala bewerten sollte. Außerdem wurde nach Schwierigkeiten während der Bearbeitung gefragt und die Gründe für diese. Aus diesen lassen sich Probleme im Design ableiten.

6.2 Durchführung

6.2.1 Population

Aus zeitlichen Gründen haben an der Evaluation insgesamt nur 4 Probanden teilgenommen. Von dieses waren zwei Informatikstudenten, wobei einer sich im Bachelor- und der andere im Masterstudium befand. Die anderen beiden Probanden sind Studenten anderer Fachrichtungen, die jedoch durch hohe technische Computerfähigkeiten dennoch als Probanden für die Sicht der Stakeholder dienen konnten. Nur die Informatikstudenten kannten sich bereits im voraus mit der Anforderungserhebung aus, jedoch sind Vorkenntnisse an dieser Stelle nicht benötigt wurden. Keiner der Probanden hat bereits Vision Videos für das Requirements Engineering erstellt und nur der Masterstudent war bereits an der Entwicklung einer Software beteiligt.

6.2.2 Ablauf einer Sitzung

Als das zum visualisierende Szenario wurde „Buchausleihe mithilfe einer Smartphone-App“ gewählt. Für dieses wurde bereits ein Großteil der Artefakte bereitgestellt, sodass der Proband nur einmal dazu aufgefordert wurde, ein neues Video mit dem Vision Video Maker aufzunehmen sowie einmalig ein existierendes Artefakt zu editieren. Auch wurde das Szenario bereits vor Beginn der Bearbeitung geöffnet und die Artefakte in das Programm geladen.

Nachdem das Ziel des Vision Video Makers erklärt wurde und nach einer Einarbeitungszeit in das Programm, welche gemessen wurde, sollte der Proband im ersten Teil die Aufgaben in der Aufgabenstellung bearbeiten. In der Einarbeitungszeit sollte sich der Proband durch das Programm bewegen, um Arbeitsweise, Bedienung und Funktionen zu testen. Weiterhin sollte sich der Benutzer in das Szenario und die bereitgestellten Artefakte einarbeiten. War der Proband bereit, so startete die Aufgabenbearbeitung. Für jede Aufgabe wurde die Zeit gemessen, bis der Proband meinte, fertig mit dieser zu seien, bevor er die nächste angefangen hat . Im Anschluss auf die Aufgaben wurde der Bewertungsbogen zusammen mit dem Probanden ausgefüllt.

6.3 Auswertung

In den folgenden Unterkapiteln werden die Ergebnisse der Evaluation präsentiert und daraufhin bewertet. Anschließend werden die Bedrohungen der Validität der Ergebnisse beschreiben.

6.3.1 Ergebnisse der Evaluation

Die gemessenen Daten werden in den Tabellen 6.1, 6.2 und 6.3 dargestellt und wurden in Sekunden gemessen. Dabei wird unterschieden in die Einarbeitungszeit des Probanden und der Bearbeitungszeit der Aufgaben.

Einarbeitungszeit

Die Einarbeitungszeit in Sekunden ist die Zeit, die der Proband damit verbracht hat, sich in den Vision Video Maker einzuarbeiten, bis dieser meinte, bereit für die Aufgaben zu seien. Dazu gehörten verschiedene Funtkionen zu testen, zum Beispiel die Menü-Optionen, eine Mockup-Testaufnahme und das Zeichnen auf einer leeren Leinwand.

	Mittelwert	**Standardabweichung**	**Median**
Einarbeitungszeit (s)	519	197,67	497

Tabelle 6.1: Mittelwert, Standardabweichung und Median der Probanden

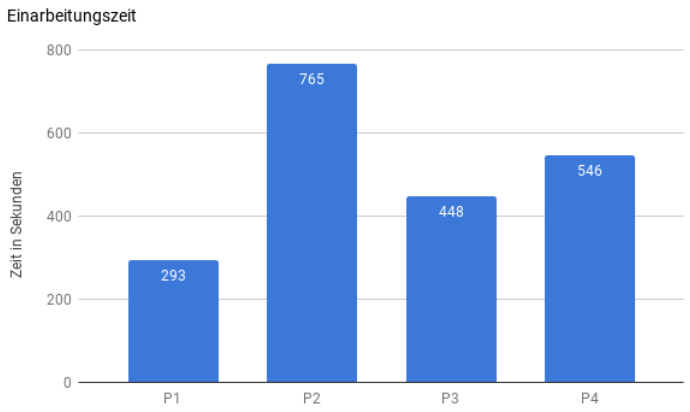

Abbildung 6.1: Einarbeitungszeit der Probanden in Sekunden. P1 und P2 sind Informatiker.

Der Mittelwerte und Median liegen dicht bei einander, im Durchschnitt hat der Proband 4:39min zur Einarbeitung in das Programm benötigt. Da es jedoch starke Unterschiede in den Zeiten gibt, lässt sich ohne weitere Probanden keine eindeutige Aussage über die durchschnittliche Einarbeitungszeit für Personen machen. Während sich ein Proband mit allen, auch nicht für die Aufgaben benötigten Funktionen vertraut gemacht hat, hat der Rest nur die Optionen kurz getestet, die benötigt wurden, wodurch es zu unterschiedlichen Zeiten kommt.

Bearbeitungszeit der Aufgaben

Die folgenden Tabellen stellen die Bearbeitungszeiten der Aufgaben in Sekunden dar. Gemessen wurde die benötigte Zeit der (Teil-)Aufgabe, bis der Benutzer angab, fertig zu seien.

Zur besseren Übersicht wurden die Tabellen in Aufgabe 1 und dessen Teilaufgaben sowie Aufgabe 2 bis 4 der Aufgabenstellung aufgeteilt.

Aufgabe	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**
Fokussierter Aspekt	Erstellung des Szenarios					
Mittelwert (s)	32,25	100	55,25	33,25	154,5	36,75
Median (s)	31.5	99	48,5	29,5	153,5	35,3
Max (s)	38	108	78	56	170	44
Min (s)	28	94	46	18	141	32
Erwartete Dauer (s)	30	90	40	30	180	30

Tabelle 6.2: Mittelwert, Median, Maximum, Minimum und erwartete Dauer für Aufgabe 1 der Evaluation

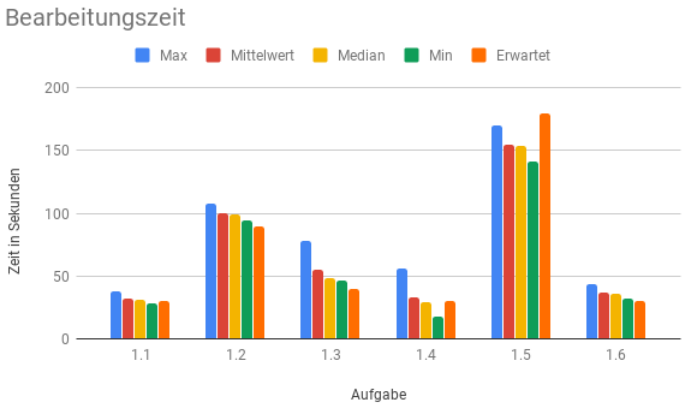

Abbildung 6.2: Bearbeitungszeit der Probanden in Sekunden für Aufgabe 1 (Visualisierung eines Szenarios mithilfe von gestellten und aufgenommenen Artefakten)

Der Mittelwert und Median der Bearbeitungszeiten für die ersten Teilaufgaben liegen nah bei einander. Jedoch gibt es zum Teil starke Unterschiede zwischen dem Maximum und Minimum. Die gemessenen Werte liegen im Durchschnitt leicht über den erwarteten Werten. Die Werte wurden im obigen Säulendiagramm 6.2 visualisiert.

Aufgabe	**2**	**3**	**4**
Fokussierter Aspekt	Editieren von Artefakten	Undefinierter Schritt finden	Export
Mittelwert (s)	64,75	65	96,75
Median (s)	59,5	58,5	97,5
Max (s)	97	93	105
Min (s)	43	50	87
Erwartete Dauer (s)	-	20	90

Tabelle 6.3: Mittelwert, Median, Maximum, Minimum und erwartete Dauer für Aufgabe 2 bis 4 der Evaluation. Eine erwartete Dauer für Aufgabe 2 wurde wegen unterschiedlichem Umfang der Bearbeitungsmöglichkeiten nicht angeben.

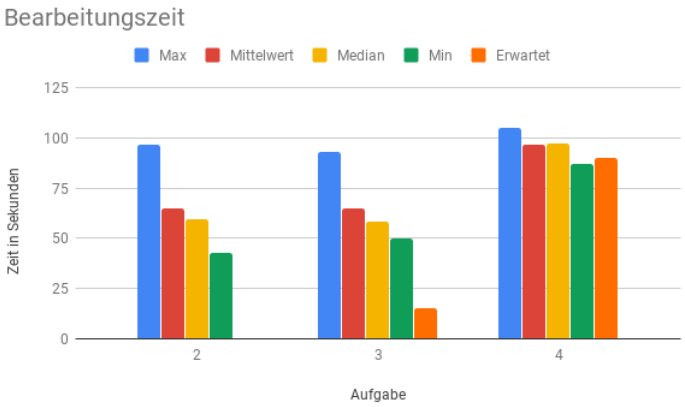

Abbildung 6.3: Bearbeitungszeit der Probanden in Sekunden für die Aufgabe 2 (Editieren eines Artefakts), Aufgabe 3 (Undefinierten Schritt hinzufügen) und Aufgabe 4 (Exportieren)

Der Mittelwert und Median liegen für Aufgabe 2 und 3 stärker auseinander, für Aufgabe 4 sind sie nahezu gleich. Die Zeit für Aufgabe 2 ist abhängig davon, inwieweit der Proband dazu bereit war, ein Artefakt zu editieren. Aus diesem Grund wurde keine erwartete Zeit angegeben, da diese Aufgabe in unterschiedlich langem Umfang ausführbar ist und nur testen sollte, ob die Probanden Mittel zum Editieren von Artefakten finden. Für diese Aufgabe haben alle Probanden die Mal-Funktion verwendet, wodurch relativ ähnliche Werte entstanden sind. Bei Aufgabe 3 hatten alle Probanden ein Problem, die Option für den undefinierten Schritt zu finden. Nach etwa einer Minute hat der Prüfer eingegriffen oder der Benutzer selber gefragt, was zu machen ist, wodurch kein Proband die Aufgabe selbst gelöst hat. Die Zeiten der 4. Aufgabe sind von der Exportgeschwindigkeit abhängig. Die Werte wurden im obigen Säulendiagramm 6.3 visualisiert.

Bewertung der Aspekte

Die letzte Tabelle die angekreuzten Antworten des Bewertungsbogen. Es werden die Anzahl der in der Liker-Skala ausgewählten Antworten zusammengezählt und präsentiert.

Aspekt	**Bewertung der Probanden**				
	sehr schlecht	**schlecht**	**neutral**	**gut**	**sehr gut**
Gesamteindruck	0	0	3	1	0
Übersichtlichkeit	0	0	2	2	0
Bedienbarkeit	0	1	0	3	0
Selbsterklärend	0	0	1	2	1
Videoqualität	0	0	4	0	0

6.3.2 Bewertung der Ergebnisse

Bearbeitungszeit

Da der Proband sich bereits in der Einarbeitungszeit mit dem Programm vertraut gemacht hat, wussten sie bereits von Beginn an, wo ein Großteil der gefragten Funktionen vorhanden ist, was dem realen Testen von Programmen vor dem Gebrauch gleich kommt. Es ließ sich jedoch kein Lerneffekt feststellen, was mit der Komplexität des Programms und der kurzen Nutzungszeit zu begründen ist. Zumindest ein Proband hat jedoch nach einiger Zeit Lerneffekte gezeigt ist und hat spätere Aufgaben in niedriger Zeit als erwartet erledigt.

Jedoch gaben die Probanden in der Evaluation auch Schwierigkeiten in der Bearbeitung der Aufgaben an.

So fiel es einem Probanden durch die limitierte Größe des Testgeräts schwer, die Artefakte aus dem Chooser in die Timeline zu ziehen. Die Option zum Hinzufügen per Klick auf den Plus-Knopf ist ihm dabei nicht aufgefallen, da die Timeline gesenkt war. Auch war ihm nicht bewusst, wie sich Tippfehler bei der Mockup-Aufnahme verhalten und wie man bei existierenden Interaktionsaufnahmen neue Interaktionen hinzufügen kann.

Besonders bei der Aufgabe 2 hatten viele Probanden das Nichtvorhandensein einer Textfunktion sowie die limitierte Auswahl an Zeichenoptionen kritisiert.

Bei Aufgabe 3 gab es die meisten Schwierigkeiten. So hat kein Proband die in der Aufgabe angedeutete Funktion zum Hinzufügen von undefinierten Schritten verwendet, wodurch nach etwa einer Minute die Aufgabe unterbrochen werden musste. Alle Teilnehmer gaben an, dass sie nicht wussten, was in der Aufgabe verlangt wurde und was die Option tatsächlich tut. So lässt sich sowohl auf Probleme durch fehlende Informationen im Programm und in einer schlecht formulierten Aufgabenstellung schließen.

Bei der Aufgabe 4 kritisierten viele Probanden das vorhanden sein der Export-Funktion an zwei verschiedenen Stellen, wobei die Gesamtvideo-Funktion schlecht gekennzeichnet wurde. Die Exportzeit selbst lag für alle im erwarteten Zeitraum.

Bewertung des Prototypen

Der Vision Video Maker hat generell neutrale bis gute Bewertungen erhalten. Vor allem die Qualität des Videos wurde einstimmig neutral gewählt, wobei die Videogröße und Resolution bemängelt wurden. Dies war zu erwarten, da die Größe bewusst reduziert werden musste, um in die Größe des Vision Video Makers zu passen. Da kein Proband die Videoqualität als schlecht bewertet hat, wurde zu mindestens das Ziel erreicht, Videos in der bestmöglichen Qualität zu produzieren. Probleme bei der Bearbeitung der Aufgaben, sowohl durch Beobachtung als auch durch Feedback der Probanden, und der daraus resultierende Frust können die eher neutrale Meinung zum Programm erklären. Dies schließt auf Probleme beim Verständnis des Programms in der kurzen Benutzungszeit und der Not nach einer Hilfsseite bei Verständnisproblemen.

Als Gesamtresultat ist das gesetzte Ziel, ein Konzept zur Erstellung von Vision Videos mit guter Usability, nur zum Teil erfüllt wurde. Es war zwar jedem Probanden möglich, das gesetzte Ziel zur Visualisierung des Szenarios zu erfüllen, jedoch hielt sich die Begeisterung der Probanden in Grenzen. Es gibt also noch einige Punkte, die verbessert werden müssen, um sowohl den Aufwand als auch den Frust zu verringern. Es ist jedoch mit eine positive Tendenz zu sehen, sodass eine Weiterarbeit des Prototypen sinnvoll ist.

In Maßnahmen für den Vision Video Maker wurden einige im Zeitrahmen noch mögliche Verbesserungen entwickelt. Weitere Verbesserungen für zukünftige Arbeiten werden im Ausblick genannt.

6.3.3 Bedrohung der Validität

In diesem Abschnitt wird auf die möglichen Bedrohungen der Validität eingegangen, welche die Gültigkeit der gefundenen Ergebnisse beeinflussen.

Conclusion Validity

Eine Bedrohung der *Conclusion Validity* stellt eine zu geringe Teilnehmerzahl dar, wodurch die gemessenen Daten eine erschwerte Schlussfolgerung zulassen, da die Ergebnisse zu stark von Ausreißern beeinflusst werden können. Weiterhin stellt der voreingenommene Prüfer und Auswerter eine Bedrohung dar, da es sich bei beiden um dieselbe Person handelt. Aus diesem Grund kann es bei der Auswertung zu unbewussten Fehlinterpretationen kommen, welche das Ergebnis ins Positive ziehen. Weiterhin durfte der Proband bei Problemen den Prüfer nach Hilfe fragen, wodurch die Bearbeitungszeit beeinflusst wurde. Zuletzt kann es auch durch *Random irrelavancies* in Form der Umgebungsgeräusche und anderen Personen im gleichen Raum zu Beeinflussungen kommen.

Internal Validity

Die *Internal Validity* kann dadurch bedroht werden, dass das Folgen von Schritten bei der Aufgabenbearbeitung zu einfach war und der Proband somit nicht zum eigenen Nachdenken angeregt wurde, an welchen Stellen welches Element am geeignetsten wäre. Sollten die Aufgabenschritte falsch oder missverständlich formuliert sein, so kann dies ebenfalls eine Bedrohung darstellen. Weiterhin wurde nur die Hauptfunktion (das Erstellen von Videos aus Artefakten) betrachtet, wodurch einige Nebenfunktionen, wie das Schneiden von Videos oder die Aufnahme von Audiokommentaren, nicht getestet wurden. Aus diesem Grund ist nicht die Usability des gesamten Programms getestet.

Da Videobearbeitung ressourcenintensive ist, kann das laufen von anderen Programmen im Hintergrund, wie die IDE zum Starten des Programms, die Bearbeitungszeit negative beeinflussen.

Construct Validity

Die *Construct Validity* kann durch durch *mono-operation bias* bedroht werden, da nur ein exemplarisches Szenario bearbeitet werden sollte, wodurch nicht die Komplexität eines realen Szenarios gegeben ist. Weiterhin besteht die Gefahr, dass durch fehlende Praxiserfahrung und Bezug zum Requirements Engineering der Probanden das Ergebnis bedroht wird. Als Probanden wurden zum Großteil Informatik Studenten gewählt. Die gewählten Nicht-Informatiker können hier die Sicht des Stakeholders einnehmen, da sie immer noch Erfahrung mit Computern hatten. Zuletzt können die Ergebnisse durch Vorwissen beeinflusst werden, da die Probanden vor der Aufgabenbearbeitung Zeit zum ausprobieren der Software hatten.

External Validity

Zuletzt kann die *External Validity* bedroht sein, da die Evaluation in einer Testumgebung stattfand, wodurch das Verhalten der Probanden verändert sein könnte. Weiterhin ist die Einarbeitung in das Szenario erschwert, da der Proband nicht vertraut mit dem Szenario ist. Andererseits ist das Szenario der Buchausleihe eine vertraute Aufgabe, die jedem bekannt sein dürfte und mit welcher sie bereits Vorerfahrung haben müssten.

6.4 Maßnahmen für den Vision Video Maker

Einige der in der Evaluation genannten Probleme konnten während der Ausfertigung dieser Arbeit noch behoben werden. So wurde eine Anleitung des Vision Video Makers als PDF-Datei hinzugefügt, welche bei der Einarbeitung in das Programm helfen soll. Dies kann durch einen Menü-Klick geöffnet werden und erklärt die Bedienung des Programmes sowie die einzelnen Menü-Einträge.

Weiterhin wird nun automatisch nach dem Laden der Video Elementchooser geöffnet, um anzuzeigen, dass die Artefakte erfolgreich geladen und im Programm vorhanden sind. Zudem wurde die Export-Funktion für das Gesamtvideo in das Export-Menü verschoben und auch als dieses benannt.

Kapitel 7

Fazit und Ausblick

In diesem letzten Kapitel wird ein abschließendes Fazit über die erarbeiteten Konzepte und die Entwicklung des Prototypen sowie den Ergebnissen der Evaluation präsentiert. Abschließend folgt ein Ausblick auf mögliche Erweiterungen und Verbesserungen sowie auf weiterführende Forschung mit diesem Prototyp.

7.1 Fazit

Ein gemeinsames Verständnis zwischen Kunden und Entwicklern zu finden ist ein Kernproblem im Requirements Engineering und kann bei Missverständnissen zu Problemen in der Softwareentwicklung führen. In vorherigen Arbeiten wurde nachgewiesen, dass Videos bei der Erhebung und Verifizierung von Anforderungen helfen können, jedoch wegen gefühltem hohen Aufwand nur selten eingesetzt werden.

Um diese Problematik zu beseitigen haben vorherige Arbeiten Konzepte entwickelt, um dem Entwickler bei der Erstellung von Videos zu helfen. In dieser Arbeit wurde ein Konzept entwickelt, welches die Programme der vorherigen Arbeiten in ein Gesamtkonzept verbindet und dies erweitert. Dieser sogenannte Vision Video Maker erlaubt es dem Entwickler, zusammen mit dem Kunden aus Videos, Mockups und Bildern ein Gesamtvideo zu erstellen, welches als Dokumentation und Referenzpunkt für den Entwickler dient sowie zur Verifikation an den Kunden geschickt werden kann.

Dafür wurden in dieser Arbeit zunächst die gegebenen Konzepte der vorherigen Arbeiten analysiert sowie deren Schnittstellen und technischen Implementationen ermittelt. Darauf basierend wurde ein Gesamtkonzept entwickelt, welches die anderen Arbeiten bestmöglich kombiniert und passende Erweiterungen hinzufügt. Das Konzept bietet dem Benutzer die Möglichkeit, Szenarien zu visualisieren und verschiedene Artefakte in mehreren Schritten zu unterteilen. So lassen sich Videos mit Bildern und Mockups in

einem gemeinsamen Zeitstrahl kombinieren, um verschiedene Ansichten zu präsentieren. Die Bilder und Mockups werden dabei automatisch in Videos konvertiert und danach zusammengeschnitten.

Anschließend wurde das Design und die Usability des Vision Video Makers in kleinem Rahmen evaluiert. So wurde die Zeit gemessen, die der Proband für die Visualisierung eines simplen Szenarios benötigt hat. Daraufhin wurden verschiedene Aspekte des Prototyps vom Probanden mithilfe einer Likert-Skala bewertet. Die Auswertung der Evaluation ergab einige Probleme bei der Usability des Prototypen und eher neutraler Videoqualität der erstellten Gesamtvideos, wodurch zwar der Aufwand zur Erstellung reduziert wurde, jedoch die existierenden Probleme für dauerhaften Einsatz gelöst werden müssen.

7.2 Ausblick

In diesem Abschnitt wird ein Ausblick auf mögliche Verbesserungen und Weiterentwicklungen des Vision Video Makers gegeben, die die Problematik verringern können.

Als Ergebnis der Evaluation wurde zwar eine Anleitung geschrieben und in das Programm implementiert, jedoch wäre es sinnvoller, den Benutzer beim ersten Start durch ein Hilfsmenü laufen zu lassen, welches auf die wichtigsten Funktionen zeigt und diese erklärt. Zusätzlich sollte das Design besser an das Surface angepasst werden, um ein leichteres Arbeiten mit Drag & Drop zu erlauben.

Weiterhin sollten die im Anhang genannten Probleme behoben werden, um die volle Funktionalität der Kamera zu erlauben. Falls möglich sollte die Resolution der Videos erhöht werden, um eine höhere Videoqualität zu erreichen.

Mögliche Erweiterungen wären zum einen die von den Probanden gewünschte Textoption für die Zeichen-Funktion, um Texteingaben zu erlauben sowie die Verbesserung der vorhandenen Textoptionen. So wurde das Löschen von Linien durch Klicken als schwierig bewertet.

Zum anderen sollte es dem Benutzer möglich sein, vom Programm intern die Artefakte in die Ordner zu laden, um den Aufwand nochmals zu reduzieren. Falls möglich sollte auch die Zeit zum Exportieren der Videos reduziert und der Export optimiert werden, um den Benutzer zu erlauben, im kurzen Zeitrahmen mehrere Szenarien zu erstellen.

Zuletzt ist eine erneute umfassendere Evaluation mit einer größeren Anzahl an Probanden und der Bewertung aller Funktionen des Prototypen zur Erfassung weiterer Meinungen und Verbesserungen notwendig. Dabei sollten nicht nur Informatik-Studenten und Studenten mit technischem Wissen einbezogen werden, sondern auch reale Stakeholder.

Anhang A

Anhang

A.1 Zustandsdiagramm Gesamtkonzept

Die farblich markierten Elemente sind zur besseren Übersicht in weitere Teildiagramme A.2 bis A.7 unterteilt.

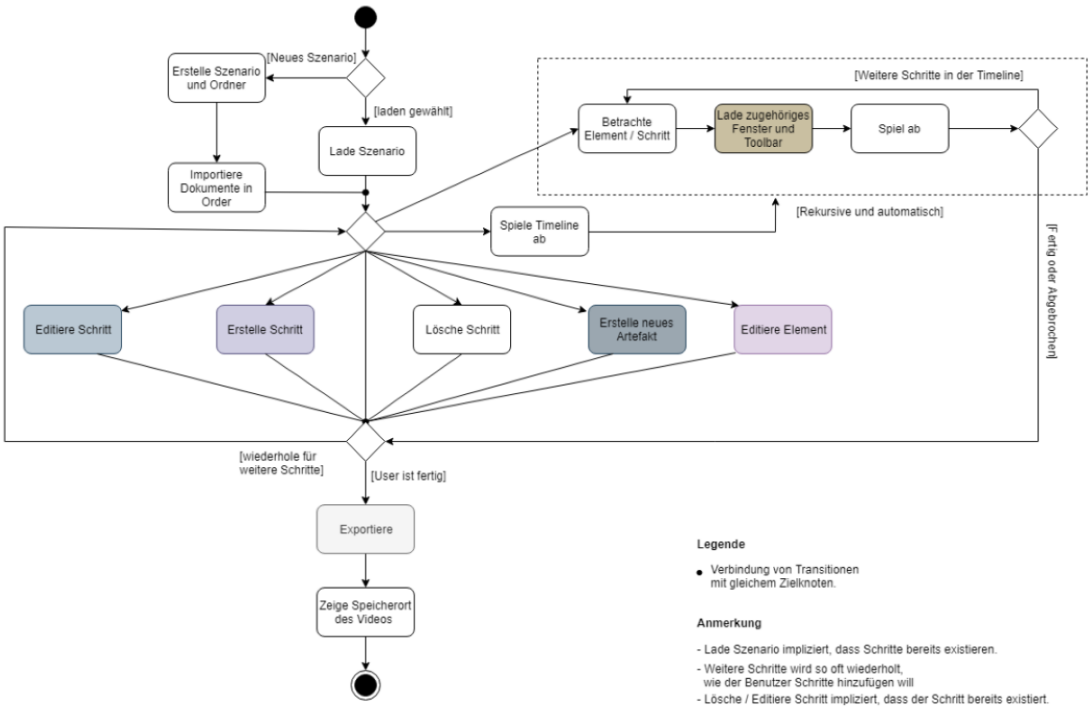

Abbildung A.1: Zustandsdiagramm Gesamtkonzept

Hinweis: In den folgenden Teildiagrammen A.2 bis A.7 gehören blau gefärbte Elemente zu Videos, rot zu Mockups und gelblich zu Bilder. Orange gefärbte Elemente gelten für Mockups und Bilder.

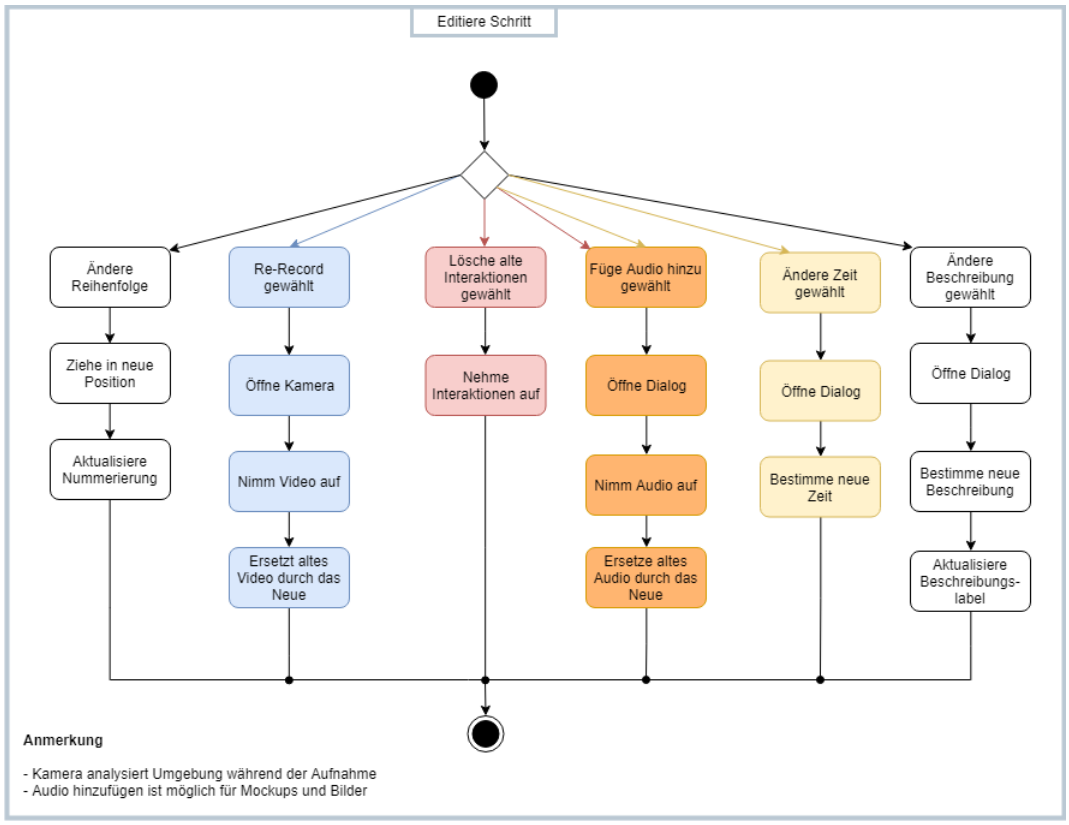

Abbildung A.2: Zustandsdiagramm zu „Editiere Schritt“

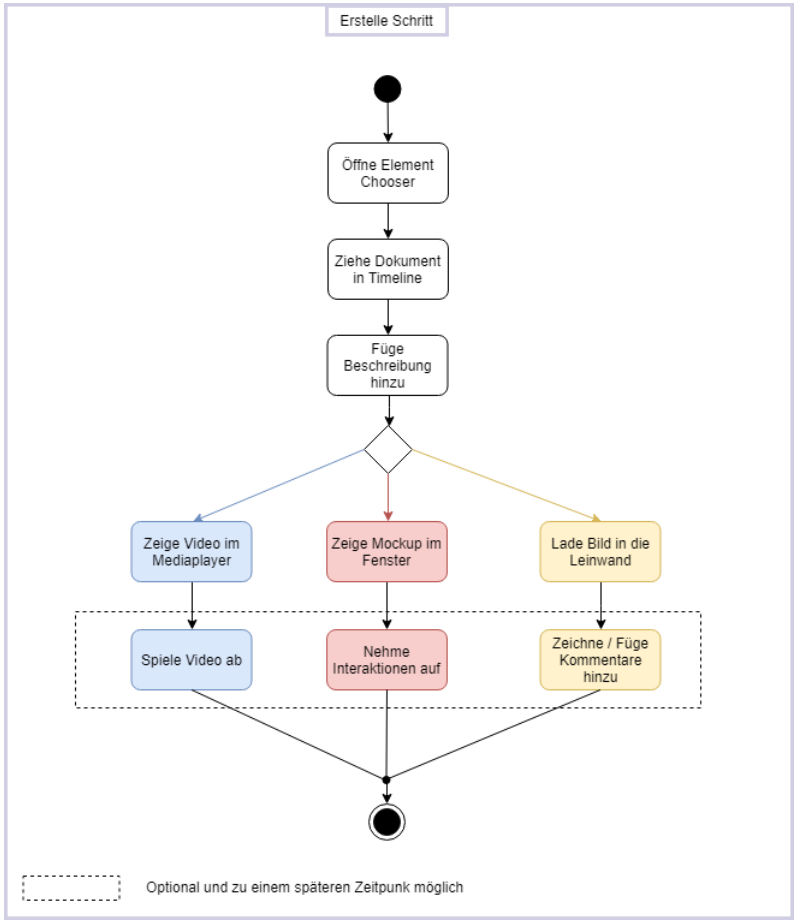

Abbildung A.3: Zustandsdiagramm zu „Erstelle Schritt“

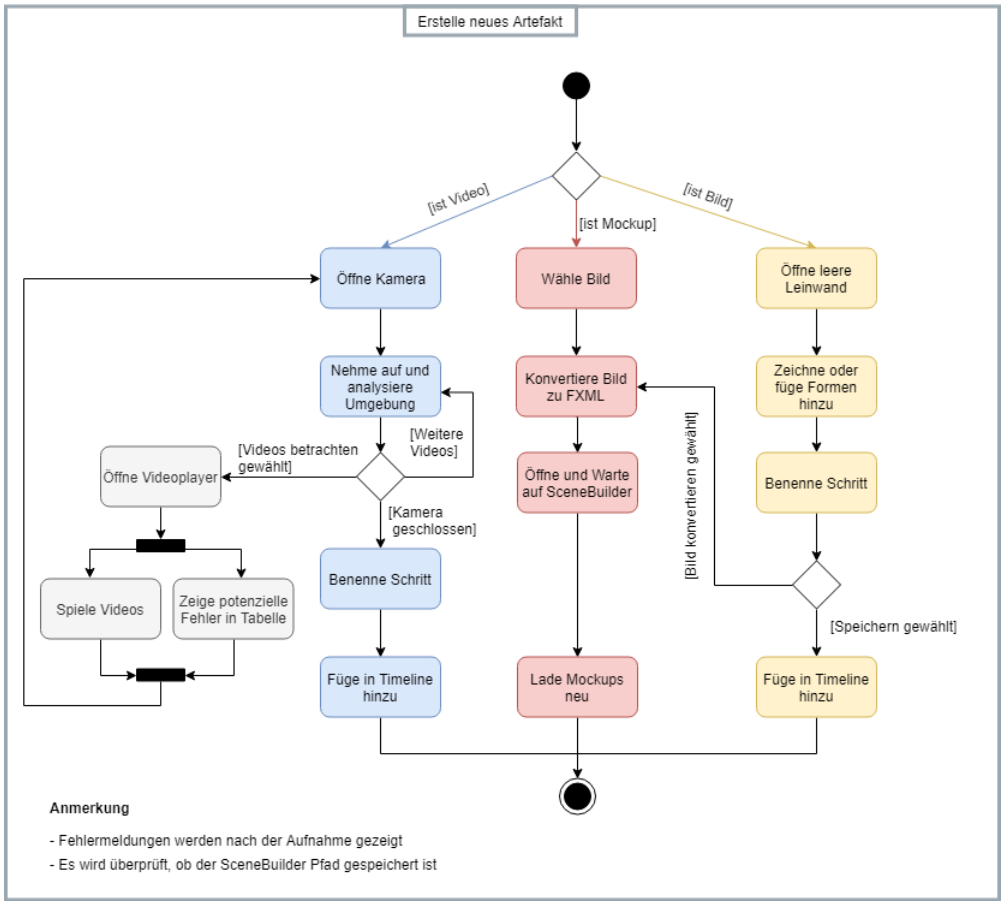

Abbildung A.4: Zustandsdiagramm zu „Neues Artefakt erstellen“

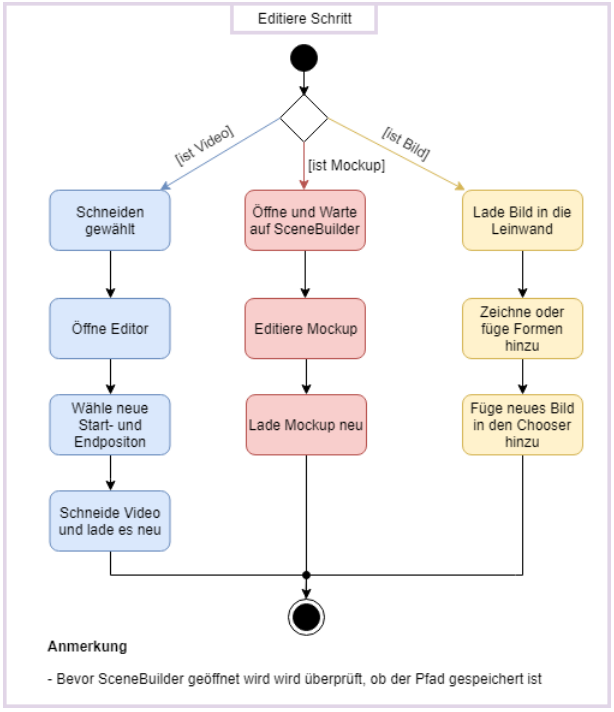

Abbildung A.5: Zustandsdiagramm zu „Editiere Element“

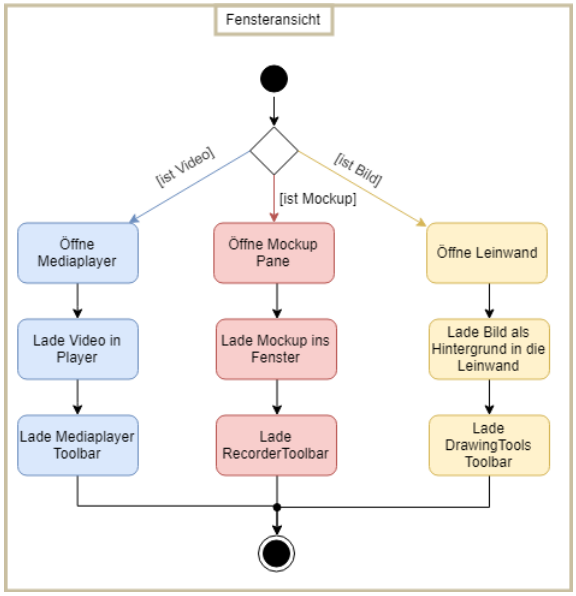

Abbildung A.6: Zustandsdiagramm zur Fensteransicht

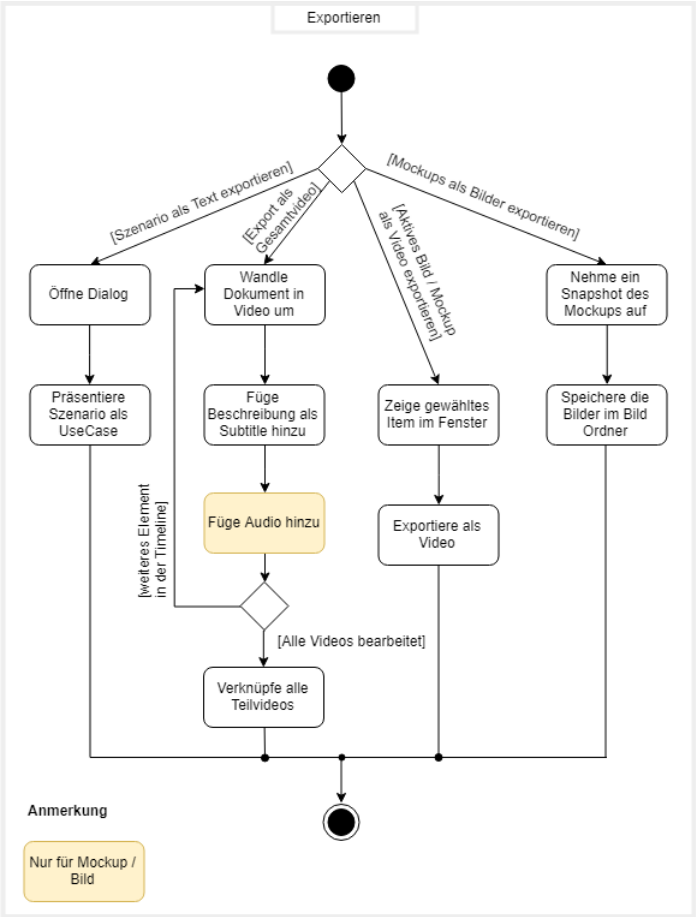

Abbildung A.7: Zustandsdiagramm zu „Exportieren“

A.2 Zustandsdiagramm ViViRecorder

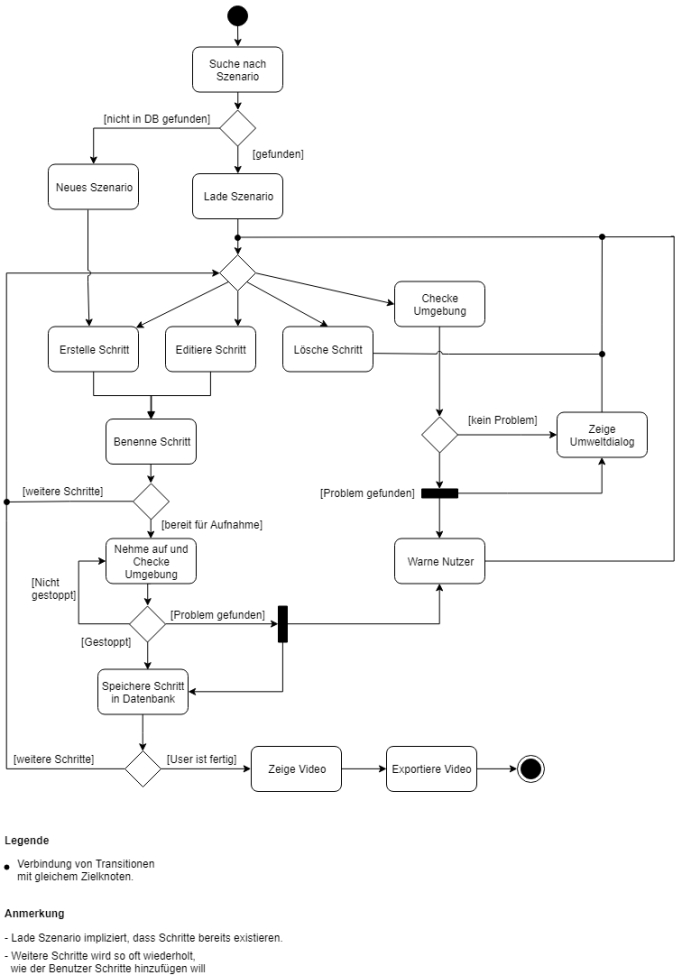

Abbildung A.8: Zustandsdiagramm des ViViRecorders

A.3 Zustandsdiagramm Mockup Recorder

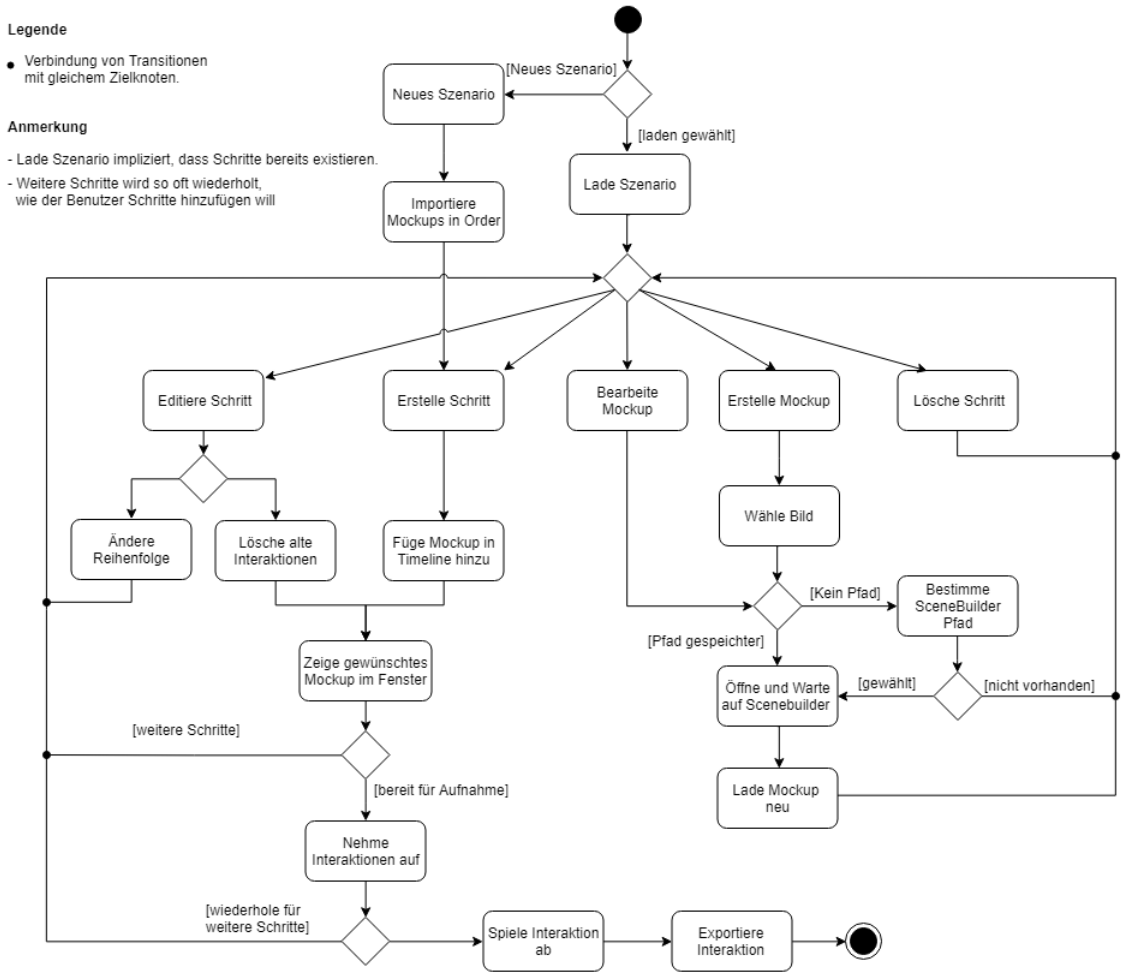

Abbildung A.9: Zustandsdiagramm des Mockup Recorders

A.4 Bekannte Probleme und Fehler

Die folgenden Probleme sind bekannt. Aus zeitlichen Gründen war es jedoch nicht möglich, alle Fehler zu beheben, da diese nicht trivial sind.

1. **Kamera**: Auf dem Surface eher seltener, auf einem Windows PC mit Webcam ein häufigeres Problem. Beim Start der Kamera besteht die Möglichkeit, dass das Programm unerwünscht terminiert. Beim Testen ist dies meist nur beim ersten Start der Kamera der Fall und im Falle eines Crashes wird eine Log-Datei von JavaCV produziert. Das größte Problem an diesem Fehler ist, dass der Benutzer seinen Fortschritt verliert. Andererseits entsteht durch Aufforderung zum Speichern bevor der Benutzer die Kamera startet die Gefahr, dass dieser weniger geneigt ist, die Kamera zu benutzen.

2. **Umweltanalyse**: Der Knopfdruck auf den Umweltanalyse-Knopf öffnet zwar das dazugehörige Dialog, jedoch wird die Analyse nie abgeschlossen. Beim Debuggen stellt sich heraus, dass die Methode zwar aufgerufen wird, die in der Methode enthaltene Schleife jedoch nie erreicht wird. Die fehlende Erfahrung im Programmieren macht es schwer, so einen Fehler zu beheben.

3. **Audio-Fehler**: Die Sensorik für Audio funktioniert nur die Hälfte der Zeit. Es ist unbekannt, wann die Sensorik funktioniert, was das Debuggen erschwert. Aus diesem Grund bekommt der Benutzer kein Feedback zu der Umgebungslautstärke. Ein möglicher Grund ist, dass die genutzte TargetDataLine nicht unterstützt wird. Da das Feature jedoch auf dem ViViRecorder funktioniert, ist dies nur eine Vermutung.

4. **Audioaufnahme**: Auch ein Fehler, der zufällig auftritt. Bei der Aufnahme von Audiokommentaren wird manchmal die letzte Sekunde des Audio nicht aufgenommen. Ein möglicher Grund ist, dass die Aufnahme gestoppt wird, während JavaCV es bearbeitet. Jedoch funktioniert die Audioaufnahme für Versionen ≤ 1.2 problemlos.

5. **Programm läuft im Hintergrund nach Starten der Kamera**: Das Programm läuft weiterhin im Hintergrund, falls der Benutzer die Kamera mindestens einmal gestartet hat. Der Grund dafür ist die jni4net Verbindung, da ohne diese das Programm auch beim Starten der Kamera normal schließt.

A.5 Inhalt der CD

In diesem Anhang befindet sich eine Auflistung der Daten auf der beiliegenden CD:

1. Bachelorarbeit in digitaler Form (PDF-Datei, LaTeX-Projekt)
2. Evaluationsunterlagen bestehend aus:
   a) Aufgabenstellung
   b) Bewertungsbogen
   c) Beispielszenario "Buchausleihe"
   d) Ergebnisse der Evaluation als Excel Tabelle und PDF-Datei
   e) Digitalisierte ausgefüllte Fragebögen
3. Quellcode des Vision Video Makers als IntelliJ-Projekt
4. Ausführbare Version des Vision Video Makers als .jar-Datei

Literaturverzeichnis

[1] S. Ambler. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process.* John Wiley & Sons, 2002.

[2] E. Börger, B. Hörger, D. Parnas, and D. Rombach. Requirements Capture, Documentation, and Validation. *Schloss Dagstuhl, Dagstuhl Report 242*, 1999.

[3] F. Brun-Cottan and P. Wall. Using Video to Re-Present the User. *Communications of the ACM*, 38(5):61–71, 1995.

[4] S. Bühne and A. Herrmann. Handbuch Requirements Management nach IREB Standard. *Aus-und Weiterbildung zum IREB Certified Professional for Requirements Engineering Advanced Level "Requirements Management". IREB eV*, 2015.

[5] J. M. Carroll and M. B. Rosson. Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems (TOIS)*, 10(2):181–212, 1992.

[6] A. Cockburn and J. Highsmith. Agile Software Ddevelopment: The People Factor. *Computer*, 34(11):131–133, 2001.

[7] O. Creighton, M. Ott, and B. Bruegge. Software Cinema-Video-based Requirements Engineering. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 109–118, 2006.

[8] C. Ebert. *Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten.* dpunkt. verlag, 2019.

[9] G. Fischer. Symmetry of Ignorance, Social Creativity, and Meta-Design. *Knowledge-Based Systems*, 13(7-8):527–537, 2000.

[10] S. A. Fricker, R. Grau, and A. Zwingli. Requirements Engineering: Best Practice. In *Requirements Engineering for Digital Health*, pages 25–46. Springer, 2015.

[11] L. Glauer. Spezifikation von Interaktionen mit graphischen Benutzerschnittstellen als Videos. Bachelor's Thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2017.

[12] J. Happe. Automatisierung von Guidelines für Videoerstellung im Requirements Engineering. Bachelor's Thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2018.

[13] S. Houde and C. Hill. What do Prototypes Prototype? In *Handbook of Human-Computer Interaction*, pages 367–381. Elsevier, 1997.

[14] M. I. Kamata and T. Tamai. How Does Requirements Quality Relate to Project Success or Failure? In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 69–78, 2007.

[15] O. Karras. Werkzeugunterstützte Analyse von Requirements-Workshop-Videos. Masterarbeit, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2015.

[16] O. Karras. Software Professionals' Attitudes towards Video as a Medium in Requirements Engineering. In *Product-Focused Software Process Improvement: 19th International Conference, PROFES 2018, Wolfsburg, Germany, 28.11 - 30.11.2018, Proceedings 19*. Springer, 2018.

[17] O. Karras, S. Kiesling, and K. Schneider. Supporting Requirements Elicitation by Tool-Supported Video Analysis. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 146–155, 2016.

[18] O. Karras and K. Schneider. Software Professionals are Not Directors: What Constitutes a Good Video? In *26th IEEE International Requirements Engineering Conference Workshops (REW)*. IEEE, 2018.

[19] O. Karras, C. Unger-Windeler, L. Glauer, and K. Schneider. Video as a By-Product of Digital Prototyping: Capturing the Dynamic Aspect of Interaction. In *Proceedings of 3rd International Workshop on Usability and Accessibility focused Requirements Engineering (UsARE 2017), RE2017*. IEEE, 2017.

[20] I. Kitzmann. Konzept und Implementierung eines Werkzeugs für multimediale Anforderungserhebung und -validierung. Masterarbeit, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2009.

[21] K. Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, Incorporated, 2010.

[22] M. Rohs. Vorlesung: Grundlagen der Mensch-Computer-Interaktion. Gottfried Wilhelm Leibniz Universität, 2017-2018.

[23] C. Rupp et al. *Requirements-Engineering und-Management: Aus der Praxis von klassisch bis gil.* Carl Hanser Verlag GmbH Co KG, 2014.

[24] K. Schneider. Vorlesung: Grundlagen der Softwaretechnik. Gottfried Wilhelm Leibniz Universität, 2016-2017.

[25] K. Schneider, O. Karras, A. Finger, and B. Zibell. Reframing Societal Discourse as Requirements Negotiation: Vision Statement. In *Proceedings of 2nd International Workshop on Crowd-Based Requirements Engineering (CrowdRE 2017), RE2017.* IEEE, 2017.

[26] N. Seyff. Collaborative Tools for Mobile Requirements Acquisition. In *Proceedings of the 19th IEEE International Conference on Automated Software Engineering*, pages 426–429. IEEE Computer Society, 2004.

[27] L. Simmet. Koevolution von multimedialen Anforderungen. Master's thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2017.

[28] I. Sommerville. *Software Engineering.* Pearson, 2018.

[29] H. Stangl. *SCRIPT: A Framework for Scenario-Driven Prototyping.* PhD thesis, Technische Universität München, 2012.

[30] H. Stangl and O. Creighton. Continuous Demonstration. In *2011 Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'11)*, pages 38–41, 08 2011.

Abbildungsverzeichnis

Tabellenverzeichnis