

HOCHSCHULE FÜR TECHNIK RAPPERSWIL

BACHELORARBEIT

**Visualisierung von Maschinendaten einer
Industriemaschine in der erweiterten
Realität**

Autoren:

Simon MÜLLER

Matthias BAUMANN

Lukas KOLLER

Betreuer:

Prof. Dr. Daniel P. POLITZE

14. Juni 2019

HOCHSCHULE FÜR TECHNIK RAPPERSWIL

Abteilung Informatik

Software

Abstract

Visualisierung von Maschinendaten einer Industriemaschine in der erweiterten Realität

von Simon MÜLLER

Matthias BAUMANN

Lukas KOLLER

Die vorliegende Bachelorarbeit beschreibt die Konzeption und Implementation einer Applikation zur Visualisierung von Maschinendaten in der erweiterten Realität. Ziel dieser Arbeit war, eine bestehende Plattform für die Verwaltung von Produktlebenszyklen so zu erweitern, dass deren Daten über eine AR-Brille visualisiert, manipuliert und persistiert werden können. Anhand einer Industriemaschine, die von einem Partner zur Verfügung gestellt wurde, konnte ein Visualisierungskonzept für die Hololens 1 entwickelt und implementiert werden. Der Fokus der Arbeit lag darin, dass der Informationsgehalt auf der bestehenden Plattform in der erweiterten Realität dargestellt wird, so dass ein Servicetechniker während der Bearbeitung von Service Cases an der Maschine vor Ort unterstützt wird.

Die Applikation bezieht die Produktdaten und zugehörigen CAD-Daten einer Instanz über einen Webservice und persistiert diese auf dem Gerät, um auch offline Daten zur Verfügung stellen zu können. Das geladene CAD-Modell wird dann als transparenter Layer über die Produktinstanz gelegt. Dies gibt dem Servicetechniker das Gefühl direkt mit der physischen Maschine interagieren zu können. Durch Gestiken wie das Anvisieren und Klicken über die Hololens können schliesslich an der Produktinstanz vor Ort virtuelle Aktionen - zum Beispiel das Auswechseln eines Bauteils - vorgenommen werden. Zusätzlich werden hilfreiche Informationen zu den Instanzen und deren Bauteile angezeigt, sodass der Servicetechniker optimal bei der Arbeit unterstützt wird.

Dank den in dieser Arbeit erzielten Ergebnissen konnte die browserbasierte Plattform für die Verwaltung von Produktlebenszyklen um einen weiteren Zugriffspunkt ausgebaut werden. Dies ermöglicht es zukünftige Servicearbeiten effizienter und zielgerichteter auszuführen und bildet eine solide Grundlage für kommende technologische Entwicklungen in der Automatisierungsbranche.

Inhaltsverzeichnis

Zusammenfassung	iii
1 Management Summary	1
2 Situationsanalyse	3
2.1 Einleitung	3
2.2 Ausgangslage	3
2.3 Ziele	4
2.4 Zusammenarbeit mit dem Industriepartner	4
2.5 Risikoanalyse	4
2.6 Projektplan	5
3 Anforderungsanalyse	7
3.1 Funktionale Anforderungen	7
3.1.1 User Szenarien	7
3.1.2 User Stories	8
3.1.3 Use Cases	8
3.2 Nicht-funktionale Anforderungen	10
3.2.1 Funktionalität	10
3.2.2 Zuverlässigkeit	11
3.2.3 Benutzbarkeit	12
3.2.4 Effizienz	13
3.2.5 Übertragbarkeit	13
4 Vorstudie	15
4.1 Evaluation	15
4.1.1 Tooling	15
4.1.2 AR Devices	16
4.1.3 Unity	19
4.1.4 Mixed-Reality-Toolkit	19
4.1.5 ARCore	20
4.1.6 Vuforia	20
4.1.7 Windows Store Deployment	20
4.1.8 Alle Werkzeuge und Tools	21
4.2 Qualitätssicherung	22
4.2.1 Code Review	22
4.2.2 Refactoring	22
4.2.3 Continuous Integration	22
4.3 Clickable UI Prototype	23
4.4 Return of Investment	24
5 Architektur	25
5.1 Architekturentscheidungen	25

5.1.1	Lokale Datenhaltung	25
5.1.2	Endpunkte im Backend	27
5.1.3	Business-Layer-Architektur	29
5.2	Domain-Model	31
5.3	Layers in der Hololens-Applikation	31
5.3.1	Layers	31
5.3.2	Business-Model	33
5.4	CAD-Daten Visitor	34
5.4.1	Repository	35
5.4.2	Bounded-Contexts	35
5.5	Anpassungen am Web-API	35
6	Learnings	39
6.1	Projektverlauf in der Retrospektive	39
6.1.1	Teilbereich: Prototyping und Tooling	39
6.1.2	Teilbereich: Backend	40
6.1.3	Teilbereich: User Interface	40
6.2	Unity	40
6.2.1	Abhängigkeiten der Applikation	40
6.2.2	Asset-Bundles	40
6.2.3	Prefabs	42
6.2.4	Unity-Web-Request	42
6.2.5	Threading	42
6.2.6	JSON-Utilties	42
6.2.7	Logging	43
6.2.8	Zenject	43
6.2.9	Testing	44
6.3	Mixed-Reality-Toolkit	44
6.4	Universal Windows Platform	45
6.4.1	Universal Windows Platform	45
6.4.2	IL2CPP-Compile-Prozess	46
6.4.3	Build auf einem Laptop	46
6.4.4	Build auf einer CAD Maschine	46
6.5	Voice	47
6.5.1	Spracherkennung	47
6.5.2	Sprachausgabe	47
7	Ergebnisse	49
7.1	Erfüllte Anforderungen	49
7.2	Nicht-funktionale Anforderungen	50
7.2.1	Funktionalität	51
7.2.2	Zuverlässigkeit	51
7.2.3	Benutzbarkeit	52
7.2.4	Effizienz	53
7.2.5	Übertragbarkeit	53
7.3	Geschwindigkeit	54
7.4	Dynamik	55
7.5	Optimierung des Speicherplatzes	55
7.6	Ladegeschwindigkeit	55
7.7	Ausbaubarkeit	56

8	Schlussfolgerung und Ausblick	57
8.1	Schlussfolgerung	57
8.2	Ausblick	57
8.2.1	Portierung zu Hololens 2	57
8.2.2	Ausbau der Hololens-Applikation	58
8.2.3	Aufarbeitung Server	58
8.2.4	Vuforia	58
8.2.5	Mixed-Reality-Toolkit	58
8.2.6	Hardware	59
8.2.7	Wirtschaftlicher Nutzen	59
	Literatur	61
A	Selbstreflexionen	63
A.1	Matthias Baumann	63
A.2	Simon Müller	64
A.3	Lukas Koller	65
B	Aufgabenstellung	67
C	Sitzungsprotokolle	73
D	Projektplan	79
E	Prototype	81
F	Architektur	93
G	Installationsanleitung	95
G.1	Voraussetzungen	95
G.2	Konfiguration und Build über Unity	96
G.3	Konfiguration und Build über Visual Studio	97
H	Administratives	99
I	Arbeit Richard Hasler	109

Abbildungsverzeichnis

2.1	Projektplan	5
4.1	GLASS Enterprise Editon [3]	17
4.2	Hololens holographic computer [5]	18
4.3	Unity Game Editor	19
4.4	Bamboo Build Task Summary	23
4.5	Prototyp Vorschau	24
5.1	Domain-Model	31
5.2	Layers der Hololens-Applikation	32
6.1	Schematische Darstellung Asset-Bundles	41
6.2	Anpassung der Build-Pipeline	41
G.1	Buildkonfiguration in Unity	96
G.2	Build-Window in Unity	97
G.3	Buildkonfiguration in Visual Studio	98

Tabellenverzeichnis

2.2	Erläuterung der Ausbaustufen	4
2.3	Risikoanalyse	5
3.2	Ausgearbeitete User Stories	8
3.5	Use Cases abgeleitet von den User Stories	10
3.7	Use Cases für AR-Erweiterungen	10
3.8	Landing Zones Genauigkeit	11
3.9	Landing Zones Fehlertoleranz	12
4.2	Anforderungskatalog an die Systeme für das Tooling	15
4.3	Systemvergleich allgemein	16
4.4	Systemvergleich enge Auswahl	16
4.7	Alle Werkzeuge und Tools	22
5.2	Evaluierte Optionen der lokalen Datenhaltung	26
5.4	Evaluierte Optionen der Backend Endpunkte	28
5.6	Evaluierte Optionen des Business-Layers	29
5.9	Relationen der Aggregatsklasse	34
5.12	Endpunkte der Serverapplikation für die Hololens	36
7.3	Umstadium der Use Cases	50
7.4	Landing Zones Genauigkeit	51
7.5	Landing Zones Fehlertoleranz	51
7.6	Landing Zones Verständlichkeit und Erlernbarkeit	52
7.7	Landing Zones Bedienbarkeit	52
7.8	Landing Zones Bildrate	53
7.9	Landing Zones Anpassbarkeit Applikationen	53
7.10	Landing Zones Technical Dept Layers	53
7.11	Landing Zones Installierbarkeit	54
7.12	Erreichte Geschwindigkeitswerte	54
G.3	Voraussetzung für Installation	96

Abkürzungsverzeichnis

AAA	Authentication Authorization Accounting
API	Application Programming Interface
AR	Augmented Reality
BOM	Bill Of Material
CAD	Computer-Aided Design
CI/CD	Continuous Integration / Continuous Delivery
CPU	Central Processing Unit
CRUD	Create Read Update Delete
CRM	Customer Relationship Model
DRY	Don't Repeat Yourself
DTO	Data Transfer Object
ERP	Enterprise Resource Planning
FPS	Frames Per Second
GUI	Graphical User Interface
HSR	HochSchule für Technik Rapperswil
HTTPS	Hypertext Transfer Protocol SSecure
IDE	Integrated Development Environment
IIS	Internet Information Service
IL	Intermediate Language
IL2CPP	Intermediate Language to C++
IPEK	Institut für Produktdesign Entwicklung und Konstruktion
MRTK	Mixed Reality ToolKit
OS	Operating System
RC	Release Candidate
SDK	Software Development Kit
SPA	Single Page Application
SSL	Secure Sockets Layer
SSML	Speech Synthesis Markup Language
TCP	Transmission Control Protocol
TFS	Team Foundation Server
UC	Use Case
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
US	Use Story
UI	User Interface
UWP	Universal Windows Platform
WAF	Windows Audio Format
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WSA	Windows Socket API
XML	Extensible Markup Language

Kapitel 1

Management Summary

Im nachfolgenden Kapitel wird die Arbeit in einer kurzen Form beschrieben, um einen Einblick zu geben wie in der Arbeit vorgegangen und welche Ergebnisse erzielt wurden. Dabei wird der Kontext des Projekts genauer erläutert.

Ausgangslage

Im Rahmen von mehreren Studien- und Bachelorarbeiten wurde bereits die Grundlage für diese Arbeit definiert. Dabei ist ein Cockpit zur Verwaltung von Maschinenlebenszyklen entstanden, das Informationen aus verschiedenen Umsystemen zusammenträgt und an einem zentralen Punkt visualisiert. Ein bereits angebundenes Umsystem ist eine Industriemaschine, die mit Sensoren ausgestattet wurde.

Ziel der Arbeit

Ziel der Arbeit ist es einen Prototypen zu entwickeln, welcher die Maschinendaten über die erweiterte Realität zugänglich macht. Das Device, das für die erweiterte Realität verwendet wird, soll seine Daten aus einem Backbone beziehen, welches durch das IPEK zur Verfügung gestellt und gewartet wird.

Vorgehen

Zu Beginn wurde zusammen mit dem Betreuer eine Aufgabenstellung erarbeitet. Diese wurde dann als Grundlage für die Risikoanalyse und den Projektplan verwendet. In einem weiteren Schritt wurden die Anforderungen genauer spezifiziert und ausgearbeitet. Zeitgleich wurden bereits erste Erfahrungen mit der Technologie und dem definierten Tooling gemacht.

Nach der Einarbeitung in die Technologien wurde ein Architektur-Prototyp erstellt, welcher die Grundstruktur der Applikation bilden sollte. Darauf aufbauend konnten schliesslich die Anforderungen entwickelt werden.

Da kein Know-how bezüglich Unity vorhanden war, musste der Prozess für die Einarbeitung und Architektur verlängert werden.

Technologien

Damit die 3D-Animationen beschrieben und die Visualisierung programmiert werden konnte, wurde Unity verwendet. Um den visualisierten Elementen ein Verhalten zuzuordnen, konnte über die Entwicklungsumgebung Visual Studio mittels C#-Code das Verhalten programmiert werden. Über IL2CPP wird der resultierende Code schliesslich in ein C++-Projekt umgewandelt.

Neben Unity und C# wurden weitere Frameworks verwendet, wie z.B. Vuforia, welches für die Entwicklung des Spatial Mappings verwendet wurde oder Zenject, das für die Dependency Injection notwendig war.

Weitere Technologien und Frameworks mit deren Evaluation und Entscheidungsgrundlage, sind im Kapitel 4 dokumentiert.

Ergebnisse

Dank den in dieser Arbeit erzielten Ergebnissen konnte die browserbasierte Plattform für die Verwaltung von Produktlebenszyklen um einen weiteren Zugriffspunkt ausgebaut werden. Dies ermöglicht es zukünftige Servicearbeiten effizienter und zielgerichteter auszuführen und bildet eine solide Grundlage für kommende technologische Entwicklungen in der Automatisierungsbranche.

Ausblick

Es wurde darauf geachtet, dass auf den erzielten Ergebnissen aufgebaut werden kann. Somit können Erweiterungen, wie neue Visualisierungskonzepte oder die Entwicklung neuer Use Cases ohne Probleme realisiert werden.

Auch die Portierung auf die Hololens 2, welche während der Arbeit von Microsoft released wurde, ist dank der flexiblen Konzeption ohne grossen Aufwand möglich. Die Portierung auf Hololens 2 wird im Kapitel 8.2.1 beschrieben.

Kapitel 2

Situationsanalyse

In diesem Kapitel werden die Einleitung, die Ausgangslage und die Ziele dieser Arbeit beschrieben. Zudem wird der Projektplan vorgestellt, welcher bereits in einer frühen Projektphase erarbeitet wurde. Die Ziele leiten sich aus der Aufgabenstellung ab, welche vom Betreuer gegengelesen und bestätigt wurde.

2.1 Einleitung

Basierend auf vorangehende Arbeiten soll die bestehende Lösung um eines der folgenden Feature erweitert werden:

- Maschinendaten in einem AR-Kontext aufbereiten und darstellen
- Bestehende Lösung optimieren und um Funktionalitäten erweitern

Diese Varianten wurden in zwei separaten Aufgabenstellungen festgehalten und dem Betreuer vorgelegt. Schliesslich wurde eine bevorzugte Variante gewählt und als Arbeit umgesetzt. Nachfolgend ist die bevorzugte Variante beschrieben. Der Vollständigkeit halber ist die zweite Variante im Anhang B zu finden.

2.2 Ausgangslage

Im Rahmen von mehreren Studien- und Bachelorarbeiten wurde bereits die Grundlage für diese Arbeit definiert. Dabei wurde ein Cockpit zur Verwaltung von Maschinenlebenszyklen erstellt, das Informationen aus verschiedenen Umsystemen zusammenträgt und an einem zentralen Punkt visualisiert. Die Umsysteme bestehen aus verschiedenen externen Systemen, wie z.B. einem ERP, einem CRM oder einer Verwaltung von Maschineninstanzen. Diese haben jeweils eine unabhängige Datenhaltung, welche in eine Persistenzschicht, dem Product Backbone, synchronisiert wird. Als Vorbereitung wurde die im Herbstsemester 2018 durchgeführte Studienarbeit „Aufbereitung und Visualisierung von Maschinendaten einer Industrie 4.0-Maschine“, welche Maschinendaten sammelt und in einem Backbone speichert, umgesetzt.

Damit wurde die Grundlage für weitere Arbeiten auf dem Gebiet der Maschinenlebenszyklen geschaffen. Eine dieser Arbeiten ist die Visualisierung von Maschinendaten über eine AR-Brille, welche die Servicetechniker vor Ort unterstützen soll.

2.3 Ziele

Ziel ist es, einen Prototypen zu entwickeln, welcher die Maschinendaten über eine AR-Brille zugänglich macht. Als AR-Brille soll die Hololens von Microsoft verwendet werden, welche die Daten aus dem Backbone bezieht. Das Backbone wird vom IPEK zur Verfügung gestellt und gewartet. Dabei soll ein User Szenario, welches im Kapitel 3 beschrieben ist, vollumfänglich umgesetzt werden. Da es sich um eine neue Technologie handelt, liegt der Fokus ebenfalls auf in der Arbeit gesammelten Erfahrungswerte, welche in der Realisierung von weiteren AR-bezogenen Arbeiten helfen sollen.

Im Vorfeld zur Arbeit wurden grobe Ausbaustufen definiert, die nachfolgend aufgeführt sind.

Bronze	Abgreifen und Visualisierung der Maschinendaten über eine AR-Brille.
Silber	Interaktion mit den Elementen, um Daten zu verändern.
Gold	Implementation einer Offline-Fähigkeit für die AR-Brille.

TABELLE 2.2: Erläuterung der Ausbaustufen

2.4 Zusammenarbeit mit dem Industriepartner

Das Cockpit und die in dieser Arbeit angestrebte Lösung sollten möglichst vielfältig einsetzbar sein. Um dennoch die Sichtweise eines Industriepartners in die Arbeit miteinzubeziehen, wurde mit der Firma Komax AG zusammengearbeitet. Diese stellte eine ihrer Industriemaschinen zur Verfügung, um einen möglichst realistischen Use-Case abbilden zu können.

Die Maschine wurde bereits in einer Vorarbeit mit Sensoren ausgestattet, welche Daten an einen Webservice schicken, um diese dann im Backbone zu persistieren.

2.5 Risikoanalyse

Da die Technologie noch relativ neu ist, wurde eine Risikoanalyse erstellt, um anfällige Risiken frühzeitig zu erkennen und bei Bedarf darauf reagieren zu können. In der nachfolgenden Tabelle ist zu beachten, dass sowohl der Schaden als auch die Wahrscheinlichkeit (Ws) mit den Noten 1 bis 4 bewertet wurden. Dabei entspricht 1 dem tiefsten und 4 dem höchsten Wert.

Beschreibung	Schaden	Ws	Risiko	Vorbeugung	Eintrittsverhalten
CAD-Daten ungenügend	2	4	8	Frühe Abklärung mit Betreuer	CAD-Daten nachzeichnen lassen
Equipment nicht vorhanden	4	1	4	Abmachen des Liefertermins	Alternative Besorgung (Ausleihe)
MRTK sehr fehleranfällig	1	4	4	Proof of Concept	Eigenimplementation
Komplexe AR Technologie	2	3	6	Vorstudie	Reduktion der Arbeitsaufträge
Spatial Recognition durch Hardware eingeschränkt	3	4	12	Proof of Concept	Abgabe ohne Spatial Recognition
Total	12	16	34		

TABELLE 2.3: Risikoanalyse

2.6 Projektplan

Bevor mit der Arbeit gestartet werden konnte, wurde zuerst ein Projektplan erstellt. Dieser wurde insgesamt in acht Phasen unterteilt, die zu einem erfolgreichen Projektverlauf führen sollen. Die Phasen wurden in einer iterativen Arbeitsweise durchgearbeitet. Zu jeder Phase sind Tickets definiert worden, welche in einer Projektmanagementsoftware verwaltet wurden. Im nachfolgenden Projektplan sind die verschiedenen Phasen in einem Zeitstrahl über die gesamte Projektdauer visualisiert. Eine grössere Version des Projektplans kann dem Anhang D entnommen werden.

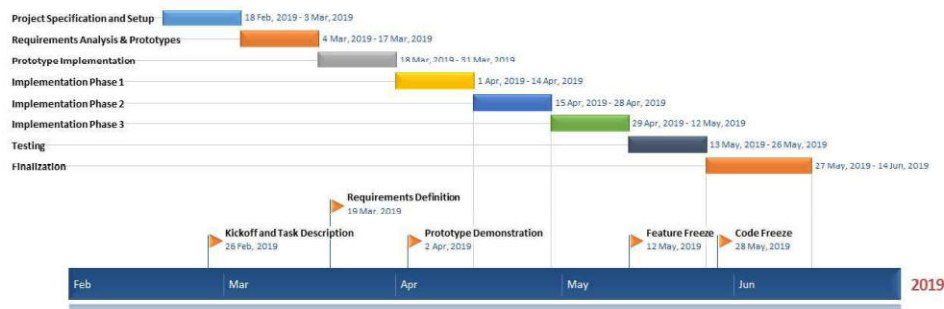


ABBILDUNG 2.1: Projektplan

Kapitel 3

Anforderungsanalyse

Im nachfolgenden Kapitel sind die funktionalen sowie die nicht-funktionalen Anforderungen, die im Rahmen dieser Thesis erarbeitet wurden, beschrieben. Diese dienen als Grundlage für die Konzeption und sind iterativ erarbeitet worden. Daraus resultierten die Use Cases, welche in drei Prioritätsstufen gegliedert sind.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen wurden zusammen mit dem Betreuer auf ihren Nutzen geprüft. Dabei wurde ein iterativer Prozess verwendet, der bei der Ausarbeitung behilflich war. Entsprechend haben sich die Anforderungen im Laufe des Projekts gemäss Wissensstand geändert. Im Folgenden sind die aktuellsten Anforderungen festgehalten.

3.1.1 User Szenarien

Folgender Szenariobeschrieb dient zur Veranschaulichung des Kontextes, in welchem sich der Servicetechniker befindet, wenn er mit der Applikation arbeitet. Das Beispiel ist frei erfunden, orientiert sich aber an einem realen Szenario.

Servicetechniker

Ein Kunde bemerkt, dass seine Industriemaschine nicht mehr korrekt funktioniert, deshalb nimmt er Kontakt mit seinem Industriepartner auf, welcher einen Servicetechniker vorbeischickt. Beim Kunden angekommen zieht der Servicetechniker seine AR-Brille an, wählt die zugehörige Maschineninstanz aus und lädt sich die Daten auf die Brille. Danach kann er die reale Maschine fokussieren und sieht auf den ersten Blick die wichtigsten Instanzdaten und aktiven Sensoren mit ihren zuletzt gelieferten Werten als erweiterte Realität auf seiner Brille. Dabei stellt er fest, dass ein Sensor unübliche Werte liefert. Dieser Sensor wird ihm in der erweiterten Realität beim Servomotor des Schwenkarms angezeigt. Mithilfe des CAD-Modells und den Zusatzinformationen, welche virtuell eingeblendet werden, baut der Servicetechniker den Servomotor auseinander. Beim Auseinanderbauen sieht er, dass das Kabel für die Stromversorgung beschädigt ist. Ebenfalls wird ihm angezeigt, wann dieses Kabel zuletzt gewartet und ersetzt wurde. Der Servicetechniker tauscht das Kabel aus und erstellt über die AR-Brille eine Modifikation, wobei diese Modifikation im

System persistiert wird. Dank der heruntergeladenen Bauanleitung kann der Servicetechniker den Servomotor wieder korrekt zusammenbauen.

3.1.2 User Stories

Die User Stories definieren Personen und deren Interesse an den Informationen. Zusätzlich beschreiben sie den Nutzen, der durch die Umsetzung der Anforderung erreicht wird. Sie leiten sich aus den User Szenarien ab und dienen als Grundlage für die Use Cases.

US01	Als Servicetechniker möchte ich die Struktur der Maschine als zusätzlichen Layer über der physischen Maschine in der erweiterte Realität sehen.
US02	Als Servicetechniker möchte ich die wichtigsten Instanzinformationen auf einen Blick sehen.
US03	Als Servicetechniker möchte ich die Service Cases zur Instanz sehen und abschliessen können.
US04	Als Servicetechniker möchte ich die wichtigsten Partinformationen auf einen Blick sehen.
US05	Als Servicetechniker möchte ich die Modifikationen auf den jeweiligen Parts sehen und erfassen können.
US06	Als Servicetechniker möchte ich die aktuellen Sensordaten auf den jeweiligen Parts sehen.
US07	Als Servicetechniker möchte ich eine Anleitung zum Service Case sehen.
US08	Als Servicetechniker möchte ich die AR-Brille ohne eine Verbindung zum Internet benutzen können.

TABELLE 3.2: Ausgearbeitete User Stories

3.1.3 Use Cases

Aus den User Stories wurden die Use Cases abgeleitet, welche in diesem Abschnitt genauer beschrieben sind. Im Gegensatz zu den User Stories sind diese detaillierter und können besser als Feature formuliert werden. Um die Use Cases zu priorisieren, wurden diese nach den Ausbaustufen Bronze, Silber und Gold gegliedert. Gold bedeutet in diesem Fall „vergoldet“ und wurde mit der kleinsten Priorität behandelt. Das Formality-Level wurde „brief“ gehalten und auf Use Case Models wurde gänzlich verzichtet.

Da die Use Cases auch zur Erstellung der Tickets im Ticketingsystem verwendet wurden, werden in diesem Abschnitt die Begrifflichkeiten verwendet, welche auch im Entwicklungsprozess Anwendung fanden. Maschinen werden als Instanzen bezeichnet und Bauteile als Parts.

Use Cases abgeleitet von den User Stories

UC01	Bronze	Instanz- und Partdaten werden dynamisch geladen.
UC02	Bronze	CAD-Modell der Instanz wird dynamisch geladen und visualisiert.
UC03	Bronze	Mapping zwischen Instanz- und Partdaten und CAD-Modell.
UC04	Bronze	Navigation durch den CAD-Modell-Baum.
UC05	Bronze	Teile der Visualisierung können aus- und wieder eingeblendet werden.
UC06	Silber	Visualisierung als Layer über der physischen Maschine (Spatial Mapping).
UC07	Gold	Bauanleitung für den Servicetechniker für einen spezifischen Service Case anzeigen.
UC08	Gold	AR-Brille kann ohne eine Internetverbindung verwendet werden.
UC09	Bronze	Instanzinformationen werden angezeigt.
UC10	Bronze	Service Cases zur Instanz werden angezeigt.
UC11	Bronze	Service Case kann geschlossen werden.
UC12	Bronze	Partinformationen werden angezeigt.
UC13	Bronze	Modifikationen zu den Parts werden angezeigt.
UC14	Bronze	Parts können gewartet werden, es wird dafür eine Modifikation erstellt werden.

UC15	Bronze	Parts können ausgetauscht werden, es wird dafür eine Modifikation erstellt werden.
UC16	Bronze	Sensordaten zu den Parts werden angezeigt.
UC17	Silber	Sensordaten werden während der Laufzeit. aktualisiert.

TABELLE 3.5: Use Cases abgeleitet von den User Stories

Use Cases für mögliche AR-Erweiterungen

UC18	Gold	Für alle Funktionen können Voice Commands verwendet werden.
UC19	Gold	Audioausgabe bei Interaktionen.
UC20	Gold	Visualisierte Elemente können manipuliert werden (Vergrößern, Verkleinern, Rotieren, Verschieben).

TABELLE 3.7: Use Cases für AR-Erweiterungen

3.2 Nicht-funktionale Anforderungen

Die nachfolgenden nicht-funktionalen Anforderungen richten sich nach der Checkliste der ISO 9126 [1].

Um die nicht-funktionalen Anforderungen vergleichbar zu machen nutzen, wird das Konzept der *Landing Zones*. [2]

Bei den Landing Zones geht es nicht um genaue Angaben, sondern um weit auseinanderliegende Kategorien, welche deutlich grösser sind als die Ungenauigkeit, in welche die Resultate zugeordnet werden können.

3.2.1 Funktionalität

Genauigkeit

Die Genauigkeit der angezeigten Maschinen- und Sensordaten wurde bereits in der Vorarbeit behandelt. Wobei die Genauigkeit der angezeigten Daten über die AR-Brille in dieser Arbeit beschrieben sind. Dabei ist zu beachten, dass es sich um die Position der angezeigten Daten handelt. Diese sollen an der richtigen Stelle der Maschine erscheinen, sprich die Partinformationen werden an der physisch korrekten Position angezeigt.

Eigenschaft	Keine Relation zu Maschine	Ungefähre Genauigkeit	Genaue Übereinstimmung
Modellposition	Modell im Raum	Modell über der Maschine	Modell genau über der Maschine
Sensordaten	Sensordaten angezeigt	Sensordaten über dem Part	Sensordaten genau beim Part
Bedienelemente	Elemente sind sichtbar	Elemente neben der Maschine	Elemente unter dem Part & immer anklickbar

TABELLE 3.8: Landing Zones Genauigkeit

Interoperabilität

Die Interoperabilität ist durch die Vorarbeit ebenfalls gegeben. Es werden alle Daten in einem Format abgelegt, mit welchem alle Applikationen arbeiten können. Diese Eigenschaft wurde im Rahmen dieser Arbeit nicht verletzt, da es sich vor allem um das Anzeigen der gegebenen Daten handelt.

Sicherheit

Die gegebene SPA kommuniziert bereits per SSL verschlüsselt über HTTPS wodurch die erforderliche Sicherheit gegeben ist. Verleih und Verwendung von AR-Brillen sind für Servicetechniker vorbehalten und sind Teil der organisatorischen Sicherheit.

3.2.2 Zuverlässigkeit

Fehlertoleranz

Da es sich bei dieser Arbeit um einen Prototypen handelt, sind Netzerkausfälle nicht durch ein Fault Tolerant Scenario abgedeckt. Im Rahmen der User Story, welche die Offline-Verfügbarkeit gewährleistet, wurde spezifiziert wie dies gehandhabt werden könnte.

Eigenschaft	Keine Toleranz	Ignorieren	Automatische Behandlung
Daten lesen	Lesen bricht ab	Lesen ignoriert fehlerhafte Daten	Code behandelt Fehler mit einem Lösungsansatz
Daten schreiben	Schreiben bricht ab	Schreiben ignoriert	Schreiben wird zwischengespeichert
Fehlerhaftes CAD	App stürzt ab	App ignoriert Fehler	App behandelt Fehler und informiert den User

TABELLE 3.9: Landing Zones Fehlertoleranz

Wiederherstellbarkeit

Die Daten, welche über die AR-Brille angezeigt werden sind im Backbone persistiert und werden von dem IPEK an der HSR verwaltet. Der Code für die AR-Brille ist in einem lokal installierten Repository gespeichert und kann jederzeit neu auf das Gerät geladen werden.

3.2.3 Benutzbarkeit

Verständlichkeit und Erlernbarkeit

Um eine verständliche Oberfläche zu gewährleisten wurde darauf geachtet, dass die angezeigten Daten auf das wichtigste reduziert sind, dadurch sind die relevanten Informationen schneller zu finden.

Da keine grundlegend neue Funktionalitäten im Cockpit bzw. in der AR-Brille zur Verfügung stehen, muss neben der Bedienung der AR-Brille nichts neues gelernt werden. Ein Tutorial, welches von der Hololens-Software zur Verfügung gestellt wird, hilft beim Erlernen der wichtigsten Gesten.

Bedienbarkeit

Über die AR-Brille werden nur Daten angezeigt, die Bearbeitung erfolgt über die SPA. Dadurch ist eine einfache und benutzerfreundliche Bedienung der AR-Brille mögliche. Funktionen der AR-Brille wurden auf die wichtigsten begrenzt. Dabei ist es möglich Daten anzuzeigen, zu verschieben, zu vergrößern und zu verkleinern.

3.2.4 Effizienz

Fliessend

Die erweiterte Realität soll ruckelfrei angezeigt werden, um Unannehmlichkeiten bei der Bedienung zu verhindern. Eine hohe Bildrate stellt dabei einen wesentlichen Faktor dar.

3.2.5 Übertragbarkeit

Anpassbarkeit

Die AR-Brille ist komplett losgelöst vom Rest der Applikation. Dadurch können Änderungen an der Brille vorgenommen werden, ohne dass diese einen Einfluss auf das Verhalten bisheriger Umsysteme haben.

Technical Dept

Ungenutzter Code, Tests und Features machen den Code und die Konfiguration komplex ohne einen Mehrwert zu bringen. Von Zeit zu Zeit muss die *Technical Dept* abgebaut werden.

Installierbarkeit

Um ein Softwareupdate auf die AR-Brille zu laden benötigt es keine vertieften Kenntnisse, da der Code direkt auf die Brille publiziert werden kann.

Kapitel 4

Vorstudie

In der Vorstudie wird die Evaluation von unbekannten Faktoren und verwendeten Tools beschrieben. Zuerst wird auf die Evaluationsphase eingegangen, in welcher vor allem die Hard- und Softwarefakten überprüft worden sind, um einen möglichst effizienten Entwicklungsprozess zu gewährleisten.

Später wird die Qualitätssicherung beschrieben, die die Grundlage für eine hohe Wartbarkeit bildet.

Zum Schluss wird noch auf den ausgearbeiteten UI Prototype eingegangen, welcher als Grundlage für die nachfolgende Realisierungsphase diene.

4.1 Evaluation

4.1.1 Tooling

Um den den Entwicklungsstand besser überwachen zu können, wurde ein Ticketing-System, ein Code Repository und ein Continous-Integration-Service verwendet.

Es wurde folgender Anforderungskatalog aufgestellt:

Ticketing	Ticketverwaltung, Kanban-Board und Backlog
Code Repository	Git, Pull Requests und Codekommentierung
Continous Integration	Build-on-push, automated Tests und Unity-Kompatibilität
Integration	Alle Services müssen zentralisiert sein
Multibenutzer-Fähigkeit	Die Services müssen für mehrere Nutzer gleichzeitig verfügbar sein
On-Premise	Die Software muss lokal installierbar sein

TABELLE 4.2: Anforderungskatalog an die Systeme für das Tooling

Mithilfe dieses Kataloges wurden folgende Systeme evaluiert:

Eigenschaft	Github	Gitlab	TFS	Atlassian
Ticketing	(✓)	✓	✓	✓
Code Repository	✓	✓	✓	✓
Continuous Integration	✗	(✓)	✓	✓
Integration	✗	✗	✓	✓
Multi-User	✓	✓	✓	✓
On premise	✗	✓	✓	✓

TABELLE 4.3: Systemvergleich allgemein

Wie aus der Tabelle entnommen werden kann, erfüllt Github die Anforderungen nicht und wird nicht weiter berücksichtigt. Gitlab kann zwar CI/CD, aber nur auf einem Kubernetes-Netzwerk, welches auf einem Windows-Server schwierig einzurichten ist. Aus diesem Grund wurde auch Gitlab nicht weiter beachtet.

Nach der Grobauswahl, wurde eine Feinanalyse für die Atlassian und TFS Systeme erstellt.

Eigenschaft	TFS	Atlassian
Windows Integration	✓	✗
Monolithisch	✓	✗
Serviceorientiert	✗	✓
Konfigurierbarkeit	✓	✓

TABELLE 4.4: Systemvergleich enge Auswahl

Aufgrund der flexiblen Konfiguration und serviceorientierten Lösung wurde die Atlassian-Produktfamilie verwendet. Diese hat den Vorteil, dass nicht genutzte Services sehr einfach abgestellt werden können.

4.1.2 AR Devices

Es gibt verschiedenste Lösungen für AR-Implementationen auf dem Markt. Diese wurden in einem ersten Schritt recherchiert, analysiert und bewertet. Da die Beschaffung der Hardware relativ teuer ist, konnte nur auf Basis von Drittbewertungen argumentiert werden. In diesem Kapitel wird eine Auswahl der verschiedenen Technologien aufgeführt und basierend auf den Informationen die Entscheidung beschrieben.

Google Glass Enterprise Edition



ABBILDUNG 4.1: GLASS Enterprise Edition [3]

Eine der wohl bekanntesten AR-Brillen ist die Google Glass. [4] Sie gehört zu den nicht-immersiven AR-Brillen und verfügt über einen Minicomputer mit Kamera und einem Display. Die Brille wurde im Rahmen eines Forschungsprojekts entwickelt und erstmals im Jahr 2014 für Privatpersonen zur Verfügung gestellt. Da ein umfangreicher Vertrieb der Brille ausblieb, wurde das Projekt bereits im Jahr 2015 wieder eingestellt. Seit 2017 wird die Brille aber in einer Enterprise Edition angeboten.

Der Vorteil der Brille ist vor allem das schlanke Design. Da sie nicht viel grösser als eine herkömmliche Brille ist, liegt diese gut auf der Nase und behindert das Sichtfeld nur gering. Dies ist auch der Grund, weshalb diese oftmals in der Fertigungsindustrie Anwendung findet.

Der Nachteil an der Hololens ist deren Gewicht und das eingeschränkte Sichtfeld von gerade mal 30°. Zudem ist die Akkulaufzeit von drei Stunden sehr kurz für den angedachten Anwendungsfall.

Microsoft Hololens



ABBILDUNG 4.2: Hololens holographic computer [5]

Die Hololens ist eine von Microsoft entwickelte AR-Brille. [6] Sie ermöglicht die Darstellung interaktiver 3D-Projektionen direkt im Sichtfeld des Anwenders. Verfügbar ist sie seit 2016 in einer Development Edition und als Commercial Suite mit zusätzlichen Funktionalitäten für Unternehmen. Eine neue Version ist im Verlaufe des Jahres 2019 geplant.

Dank der einzigartigen Technologie bietet die Hololens die Funktionalität einer echten erweiterten Realität. Im Gegensatz zur Google Glass ist kein Umweg über die Kamera nötig, um AR-spezifische Elemente anzeigen zu können.

Der Nachteil an der Hololens ist deren Gewicht und das eingeschränkte Sichtfeld. Mit nur gerademal 30 ° horizontalem Sichtfeld ist dieses noch sehr eingeschränkt. Zudem ist die Akkulaufzeit mit drei Stunden etwas knapp bemessen für den branchenspezifischen Anwendungsfall.

Entscheidung

Es wurde entschieden, dass die Applikation für die Hololens 1 implementiert wird. Dies weil die Technologie der Microsoft Hololens einzigartig und eine zweiten Version in der Produktion ist. Desweiteren fiel eine Beschaffung der ersten Version weg, da bereits Exemplare von der HSR zur Verfügung gestellt werden konnte. Es wurde dabei grossen Wert auf Upgradefähigkeit auf die zweite Version gelegt.

4.1.3 Unity

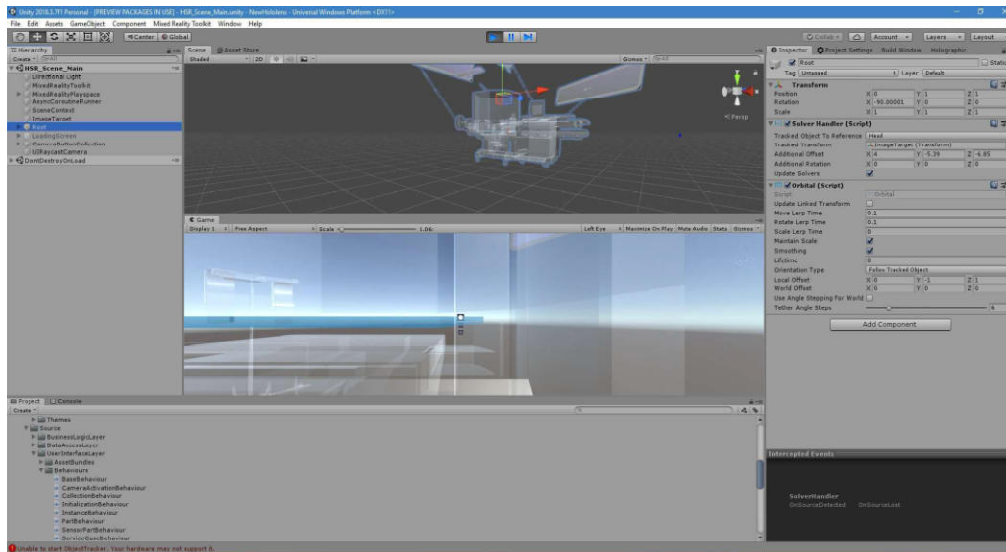


ABBILDUNG 4.3: Unity Game Editor

Unity ist eine Laufzeit- und Entwicklungsumgebung für 3D-Animationen und findet vor allem in der Spieleentwicklung Anwendung. [7] Aber auch für die Anwendung jenseits von Computerspielen ist Unity bestens geeignet. So können AR-Inhalte für die Hololens erzeugt und über die Brille gerendert werden.

Unity wird ausserdem auch von Microsoft als Entwicklungsumgebung für die Implementierung von Mixed-Reality-Anwendungen empfohlen.

4.1.4 Mixed-Reality-Toolkit

Das Mixed-Reality-Toolkit, kurz MRTK, ist eine Erweiterung für die Unity Plattform und wird von Microsoft zur Verfügung gestellt. [8] Es wurde speziell für die Realisierung von Mixed-Reality-Applikationen entwickelt, unter welche auch Hololens-Applikationen fallen.

Das Toolkit kann als sogenanntes Asset in den Unity Editor importiert werden. Es erweitert den Unity Editor um eine Registerkarte für die Konfiguration von Mixed-Reality-Applikationen und ermöglicht den Zugriff auf ein von Microsoft entwickeltes SDK.

Das SDK verfügt über Hilfsklassen, welche das Implementieren von Mixed-Reality-Applikationen vereinfachen soll. So steht zum Beispiel das Script *Interactable.cs* zur Verfügung, welches einem Game-Object zugewiesen werden kann und so die holographische Interaktion mit diesem ermöglicht.

Zusätzlich zum eigentlichen MRTK kann auch ein Asset mit Beispielprojekten bezogen werden.

4.1.5 ARCore

ARCore ist ein SDK zur Entwicklung von AR-Apps für das Android Betriebssystem. [9] Es stellt Werkzeuge zur Verfügung, die den Entwickler bei der Realisierung von AR-Apps unterstützen. Somit wird die Verwendung der drei Technologien *Motion Control*, *Environmental Understanding* und *Light estimation* erheblich vereinfacht.

Im Rahmen der Arbeit ist dieses SDK deshalb interessant, weil dadurch ein weiterer Nutzungskanal geschaffen werden kann, zum Beispiel über ein handelsübliches Smartphone oder Tablet.

4.1.6 Vuforia

Vuforia ist eine Bibliothek für die Bilderkennung in der erweiterten Realität. Sie ist als Plugin in Unity verfügbar, über den Editor konfigurierbar und installiert alle notwendigen Teile in das Projektverzeichnis.

Die Bibliothek bietet verschiedene Targets in Form eines Bildes, an welche das Modell angebunden werden kann. Ausserdem kann es das Tracking weiterführen, auch wenn die Sicht auf das Target verloren gegangen ist.

Die Ausführung eines Beispielprogrammes unter der Hololens hat befriedigende Ergebnisse erzielt und die Bibliothek hat eine hohe Erkennungsrate in einem Prototypentest.

Im Rahmen dieses Projektes wurde Vuforia verwendet, um das Maschinenmodell auf die Maschine zu mappen. Dabei wird ein Bild an die physische Maschine geheftet und mit der Brille fokussiert. Sobald Vuforia das Bild erkennt, wird die Maschineninstanz umpositioniert und an der entsprechenden Position dargestellt.

4.1.7 Windows Store Deployment

Die Veröffentlichung der Applikation erfolgt normalerweise über den Windows Store. Da sie jedoch vertrauliche Daten enthält (CAD-Modell) und der Windows Store einen öffentlichen, weltweiten und bezahlten Service darstellt, wurde auf die Veröffentlichung verzichtet. [10]

Zwar könnte die Applikation einer eingeschränkten Gruppe von Usern zur Verfügung gestellt werden. [11] Jedoch wären die schützenswerten Daten trotzdem auf den Servern von Microsoft gespeichert. Für eine sichere Verteilung müsste eine HSR-interne Instanz des Microsoft Stores zur Verfügung gestellt werden. [12]

4.1.8 Alle Werkzeuge und Tools

Unity 2018.3.7f1	wird verwendet, damit die 3D-Animationen beschrieben und die Visualisierungen programmiert werden können.
Visual Studio 2017	verwaltet das C#-Projekt und ermöglicht die Implementation der Behaviour-Scripts.
IL2CPP	Aus dem Visual Studio Projekt wird schliesslich mittels IL2CPP ein C++ Projekt generiert, welches dann kompiliert und auf die Brille geladen werden kann.
Hololens Emulator	emuliert die Hololens auf dem lokalen Computer. Dies erlaubt einen schnelleren Build-Test-Zyklus und beschleunigt die Entwicklung der Applikation.
Mixed-Reality-Toolkit v2.0.0 RC1	ist eine Sammlung von Utilities für den Unity Editor, welche die Programmierung von Hololens-Applikationen vereinfacht.
Holographic Remoting Player	ist eine UWP-Applikation, die den Szenenkontext aus Unity direkt auf die Hololens projiziert.
Windows SDK und UWP SDK	stellt die Build-Chain zur Verfügung.
Latex	wird für die Dokumentation verwendet. Hierfür wurde eine der gängigen Latex-Distributionen gewählt.
Visual Studio Code	als Latex-Editor. Visual Studio Code wurde um die Extensions <i>LaTeX Workshop</i> und <i>Rewrap</i> erweitert.
Draw.io	zur Erstellung von Diagrammen.
Bitbucket	ist eine On-Premise-Installation, die als Source-Verwaltungssystem dient.
Jira	ist eine On-Premise-Installation zur Verwaltung von Tickets.
Confluence	ist eine On-Premise-Installation für Dokumentationszwecken innerhalb des Teams.

Bamboo	ist eine On-Premise-Installation als Build-Server für automatisierte Builds und Tests.
Tempo	als Erweiterung zu Jira, sodass eine Zeiterfassung zur Verfügung steht.

TABELLE 4.7: Alle Werkzeuge und Tools

4.2 Qualitätssicherung

4.2.1 Code Review

Für jedes Ticket wurde ein neuer Branch auf Basis des Masters erstellt. Sobald alle Anforderungen eines Tickets umgesetzt waren, wurde ein Pull-Request für diesen Branch erstellt. Dies hatte zur Folge, dass jede Änderung durch einen zweiten, unabhängigen Entwickler angeschaut werden musste. Das Feedback zum Pull-Request wurde, falls nötig, bilateral besprochen.

4.2.2 Refactoring

Refactorings wurden unabhängig von der Entwicklung der Anforderungen durchgeführt. Dabei wurde neben anderen gelernten Best Practices, speziell auf eine lose Kopplung zwischen Klassen und das Single Responsibility Principle [13] geachtet.

4.2.3 Continuous Integration

Mit der Wahl der Atlassian-Produktfamilie wurde im CI/CD auf den Bamboo-Server gesetzt, der eine gute Integration in die restlichen Atlassian-Services bietet.

Auf Wunsch des Betreuers wurde eine produktive Umgebung und eine Entwicklungsumgebung aufgesetzt. Bei beiden Umgebungen wurde ein CI/CD Prozess eingeführt, welcher beim Mergen von Pull Requests auf die jeweiligen Umgebungen einen Neustart der Services forcierte.

Bamboo Konfiguration

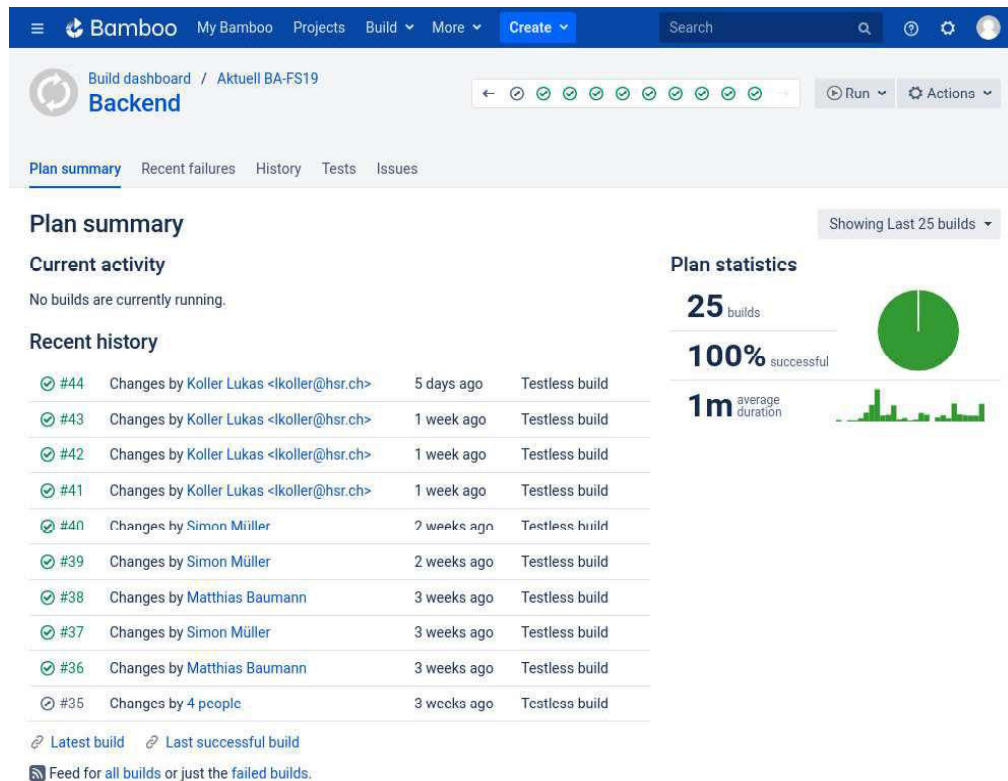


ABBILDUNG 4.4: Bamboo Build Task Summary

Das Web-Dashboard wird mithilfe von NodeJS auf dem Server gebuildet und dann auf einer IIS-Website veröffentlicht.

Auch der Backendservice wird durch continuous integration gebaut. Der Continuous Integration Server zeigt unter 4.4 Detailinformationen zu dem Backendservice Build Task.

4.3 Clickable UI Prototype

Zu Beginn der Arbeit wurde ein UI Prototype erstellt. Dieser hat dabei geholfen zu erkennen, wie sich die Applikation ungefähr verhalten soll. Dadurch konnten erste Navigationsprozesse und UI Elemente visualisiert und mit den beteiligten Stakeholdern überprüft werden.

Es gibt diverse Videos über AR Applikationen im Netz. Die meisten sind aber über Videobearbeitungsprogramme erstellt worden und existieren nicht als funktionierende Applikation. Erkennbar ist dies daran, dass das Sichtfeld weit mehr als 30 ° beträgt, wobei die Hololens 30 ° abdeckt. Dies ist jedoch ein essenzieller Teil in der Visualisierungskonzeption.

Aufgrund der Ideen und Konzepte der AR und den erwähnten Videos wurde die umsetzbaren Elemente ausgearbeitet. Da zu diesem Zeitpunkt noch kein Know-how über die Hololens und das MRTK vorhanden war, wurde der Prototyp bewusst noch sehr einfach gehalten. Während der Arbeit wurde die Applikation durch weitere Erkenntnisse dynamisch geändert und um mehr AR-Funktionen erweitert.

Nachfolgend ist eine Ansicht eines Parts des Prototypen dargestellt. Der komplette Prototyp ist im Anhang E zu finden.



ABBILDUNG 4.5: Prototyp Vorschau

4.4 Return of Investment

Während der Arbeit wurde mit dem Masterstudent Richard Hasler zusammengearbeitet für eine Return of Investment Aufstellung. Der Student erstellte aus unseren Erfahrungen im Bereich Visualisierungen eine Studie über die Rentabilität der Entwicklung einer solchen Anwendung. Die vollständige Studie kann aus dem Anhang I entnommen werden.

Die Arbeit untersucht die Beweggründe für die Umsetzung einer Virtual Reality Applikation. In einer späteren Sektion wurde der Use Case aus Sicht der Firma beschrieben. Gegen Ende wurde eine Kostenaufstellung erstellt von welcher man die Implementationskosten einer Anwendung in unserem Anwendungsgebiet ableiten kann.

Das Ergebnis der Arbeit war, dass die Kosteneinsparungen die Entwicklungskosten nicht abdecken können. Aus diesem Grund ratet der Autor einer Firma von einer solchen Investition ab

Die Entwicklungsgruppe geht davon aus, dass die Kosten einer solchen Applikation in Zukunft sinken werden.

Kapitel 5

Architektur

In diesem Kapitel ist die Architektur beschrieben und die Entscheidungen, die dazu geführt haben. Es wurden lediglich die Änderungen zu dem bisherigen Stand festgehalten. Alle bisherigen Architekturbeschriebe und deren Entscheidungen sind in separaten Dokumenten zu den Vorarbeiten dokumentiert.

5.1 Architekturentscheidungen

Zur Dokumentation von Architekturentscheidungen wird die Vorlage *madr Projekt*[14] eingesetzt. Dieses Format ist eine gängige Vorlage für die Formatierung von Architekturentscheidungen und war in der Vergangenheit ein guter Kompromiss zwischen Kürze und Informationsgehalt.

5.1.1 Lokale Datenhaltung

Status beschlossen

Entscheidungsträger Projektteam

Datum 05. März 2019

Kontext und Problemangabe

Es soll eine lokale Datenhaltung implementiert werden, damit der Servicetechniker ohne Internetzugang mit den Daten arbeiten kann. Auf welchem Gerät sollen die Daten lokal zwischengespeichert werden?

Treibkräfte für die Entscheidung

- Die Speichergrösse verschiedener Geräte, sprich wieviele Datensätze zwischengespeichert werden
- Verbindungen, die das Gerät bietet
- Zusatzfunktionen, die durch die Lösung angeboten werden können.

Evaluierte Optionen

OP1	Android-Applikation
OP2	Hololens-Applikation
OP3	Android-Applikation für Tablet und Hololens-Applikation, um gegebenenfalls Daten zu synchronisieren

TABELLE 5.2: Evaluierte Optionen der lokalen Datenhaltung

Chancen und Risiken von Option 1

Android-Applikation mit AR Möglichkeiten.

Speichergrösse 16 GB - 64 GB

Verbindungen Bluetooth, WLAN, WAN

Bedienbarkeit Sehr gut, schnell, gewohnt

- + Änderbar
- + Automatisches Nachladen sobald Internetzugang vorhanden ist
- Ungetestete Plattform für AR
- Zu kurze Batterielaufzeit, da starke Prozessornutzung durch Videoaufnahme
-> Bildbearbeitung -> Anzeigenschleife
- Keine freien Hände, da das Gerät gehalten werden muss

Chancen und Risiken von Option 2

Hololens-Applikation verbindet sich direkt mit dem Backend.

Speichergrösse 64 GB, wobei 32 GB nutzbar

Verbindungen Bluetooth, WLAN

Bedienbarkeit Schlecht - Mittel

- + Getestete Plattform
- + Freihändig
- Kein WAN
- Enge Kopplung der Backend API zur Hololens-Applikation

Chancen und Risiken von Option 3

Die Hololens hat eine ständige Verbindung zum Backend und kann Daten nachladen. Daten können zur lokalen Zwischenspeicherung auf die Android-Applikation geladen werden. Die Hololens muss den Unterschied zwischen Android-Daten und Live-Daten nicht kennen und fokussiert sich einzig auf die Anzeige und Nutzerinteraktion.

Speichergrösse 32 GB (Hololens) + 32 GB (Phone)

Verbindungen Bluetooth, WLAN, WAN

Bedienbarkeit Sehr gut, schnell, gewohnt

- + Updatebar
- + Automatisches Nachladen sobald ein Internetzugang vorhanden ist
- + Freie Hände
- + Keine enge Kopplung der Backend API zur Hololens-Applikation
- + Geringer Energieverbrauch
- Zeitaufwand
- Zwei Geräte notwendig

Entscheidung

In der Besprechung vom 05. März 2019 wurde zusammen mit dem Betreuer entschieden, dass der Prototyp mit der Option 2 umgesetzt wird.

In einem späteren Schritt könnte die Mobile-App zwischen Hololens und Backend gelegt werden. Dies erlaubt die separate Entwicklung von Hololens und Backend. Als guter Nebeneffekt können die positiven Eigenschaften der Option 1 auch in Option 3 realisiert werden.

5.1.2 Endpunkte im Backend

Status beschlossen

Entscheidungsträger Projektteam

Datum 05. März 2019

Kontext und Problemangabe

Die Serverapplikation besitzt Endpunkte, um die Daten für das Cockpit bereitzustellen.

Sollen für die Hololens-Applikation eigene Endpunkte implementiert werden oder die vorhandenen für beide Applikationen genutzt werden?

Treibkräfte für die Entscheidung

- Arbeitsaufwand
- Komplexität
- Anzahl Requests, die nötig sind, um die Daten zu laden
- Offline-Verfügbarkeit

Evaluierte Optionen

- | | |
|------------|---|
| OP1 | Gemischte Endpunkte für Web und Hololens |
| OP2 | Die Hololens-Applikation besitzt eigene Endpunkte |

TABELLE 5.4: Evaluierte Optionen der Backend Endpunkte

Chancen und Risiken von Option 1

Die vorhandenen Endpunkte für das Cockpit werden erweitert, um die Datenübertragung zur Hololens zu ermöglichen.

- + Arbeitsaufwand ist kleiner als bei Option 2
- Komplexität der Server Applikation steigt, da die selben Endpunkte für das Cockpit und die Hololens verwendet werden
- Mindestens sechs Requests (Instanz-Daten, Bom, Service Cases, Instanz, Instanz-Parts, etc.)
- Offline-Fähigkeit ist schwierig zu realisieren, da diese eine andere Lade-Logik verlangt, als das Laden vom HTTP-Server

Chancen und Risiken von Option 2

Getrennte Backend Endpunkte für die Hololens-Applikation.

- + Komplexität auf dem Server geringer, da pro Domain spezifische Endpunkte definiert sind
- + Wenige Requests, die sinnvoll getrennt sind und einfach zwischengespeichert werden können
- + Offline-Verfügbarkeit ist einfach vorhanden
- Höherer Arbeitsaufwand als bei Option 1

Entscheidung

In der Besprechung vom 05. März 2019 wurde entschieden, dass der Prototyp mit der Option 2 umgesetzt wird.

5.1.3 Business-Layer-Architektur

Status beschlossen

Entscheidungsträger Projektteam

Datum 30. April 2019

Kontext und Problemangabe

Die Hololens-Anwendung besitzt einen Business-Layer, der die Daten von den verschiedenen Endpunkten verknüpft und diese dem UI-Layer bereitstellt.

Diese Entscheidung betrifft die Varianten der Umsetzung des Business-Layer.

Treibkräfte für die Entscheidung

- Komplexität des Layers
- Komplexität der Datenabfrage im UI-Layer
- Passende API für die Entscheidung vom 05. März 2019
- Anzahl Caching-Stufen

Evaluierte Optionen

OP1 Traditionelle Layerarchitektur

OP2 Domain-Driven-Design Architektur

TABELLE 5.6: Evaluierte Optionen des Business-Layers

Chancen und Risiken von Option 1

Eine traditionelle Schichtenarchitektur lädt die Daten über alle Schichten der Applikation. Nachteilig ist, dass der UI-Layer keine Relationen zwischen den Datensätzen erhält und diese manuell auflösen muss. Zum Beispiel bekommt der UI-Layer ein DTO für ein Instance-Part und muss dann den Business-Logic-Layer aufrufen, um die dazugehörigen Modifications zu bekommen.

Diese Architektur kann nur auf Anfragen reagieren. Es ist dadurch schwierig die Daten vorab ins Memory zu laden und dann die Datenhaltung im Memory zu realisieren.

- + Komplexität des Layers ist klein, schnelle Implementation möglich
- + Passende API für das Backend möglich
- + Caching-Stufen: Durch Dictionaries können alte Abfragen gespeichert werden
- Komplexität der Abfrage ist gross da der UI-Layer die Zusammenhänge der Daten selbständig auflösen muss

Chancen und Risiken von Option 2

Eine Domain-Driven-Design-Architektur, welche die Business-Logik innerhalb des Domain-Models darstellt. Zuerst muss über ein Model-Driven-Design-Ansatz das Domain-Model im Code abgebildet werden. Der Code, der in den Domain-Klassen eingebaut ist, erstellt dann ein vollständiges Modell aller Domaindaten. Dieses Modell bildet alle Daten mithilfe von Objektassoziationen ab. Der UI-Layer kann über die Assoziationen alle zusammenhängende Daten erhalten.

- + Passende API für das Backend
- + Caching-Stufen: Das Modell ist in-memory vorhanden und alle Abfragen beziehen sich auf dieses
- + Komplexität der Abfrage ist gering, Programmierer können das Domain-Model als Referenz zur Schnittstelle verwenden
- Komplexität des Layers ist mittel, der Aufbau des Domain-Models erfolgt in-memory

Entscheidung

Aufgrund der Vorteile des Domain-Driven-Designs wurde diese Variante umgesetzt. Dies hat ebenfalls den Vorteil, dass der Aufwand im UI-Layer minimiert werden kann.

5.2 Domain-Model

Das Domain-Model zeigt die Relationen zwischen den Business-Layer-Klassen und den gespeicherten Daten innerhalb der Applikation auf. Dieses Modell wurde schliesslich verwendet, um die Business-Layer-Klassen und ihre Relationen aufzubauen.

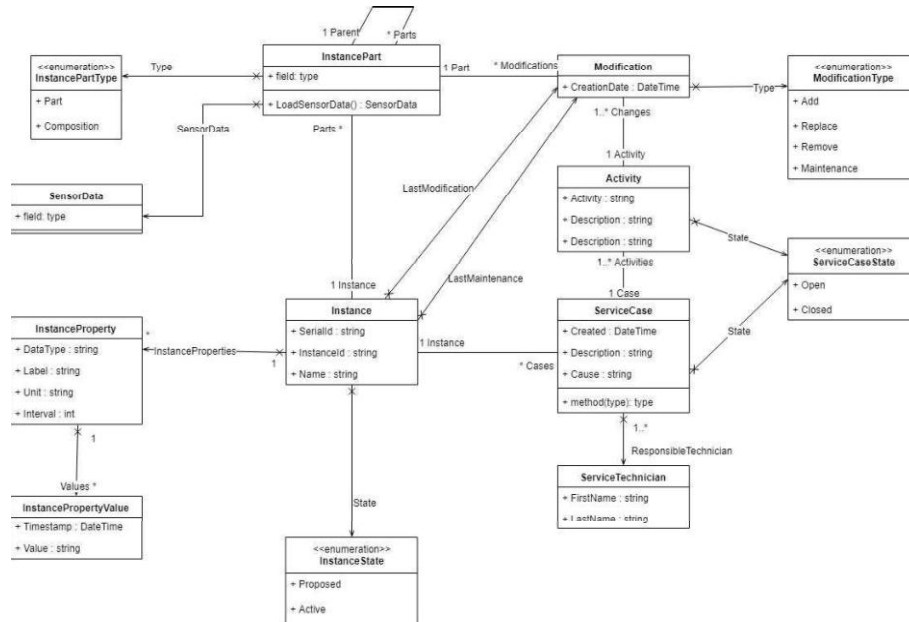


ABBILDUNG 5.1: Domain-Model

5.3 Layers in der Hololens-Applikation

5.3.1 Layers

Die Hololens-Applikation ist in drei Layers unterteilt.

1. UI-Layer
2. Business-Logic-Layer
3. Data-Access-Layer

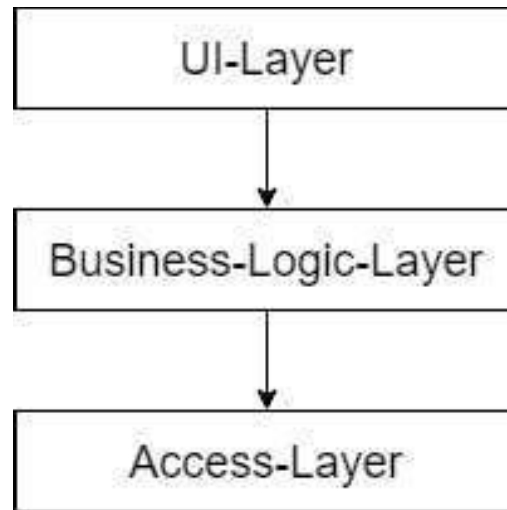


ABBILDUNG 5.2: Layers der Hololens-Applikation

Ein Layer ist eine Code-Strukturierungsmassnahme, bei welcher die Abhängigkeiten der Code-Files nur immer zum unteren Layer ziehen. Der untere Layer verlangt nie Informationen über den darüberliegenden Layer. Somit können zyklische Referenzen vermieden werden.

Layers dienen auch dem Abstrahieren von Code. Der Data-Access-Layer besitzt nur drei Klassen, die von den oberen Layers benutzt werden. Somit kann der Code innerhalb des Layers ohne grosse Auswirkungen geändert werden. Auf der Gegenseite sollte genauso versucht werden, das API des darunterliegenden Layers möglichst selten zu verwenden.

Letztlich wird versucht in einem Layer die Komplexität zu reduzieren. Zum Beispiel weiss der Business-Layer nicht wie auf eine Ressource zugegriffen wird, sondern übergibt einfach einen Uniform-Resource-Identifier (URI), wobei der Access-Layer selbständig entscheidet, woher die Ressourcen geladen werden sollten.

User-Interface-Layer

Im UI-Layer befinden sich Behaviour-Skripts, welche über die API des Business-Logic-Layers in Echtzeit Daten laden und anzeigen.

Dieser Layer wird spezifisch für Unity geschrieben und kann nicht portiert werden. Die darunterliegenden Layers sind portierbar und können auf einer anderen Plattform, wie zum Beispiel Xamarin[15] eingesetzt werden. Diese Taktik verringert die Wartungs- und Entwicklungskosten für Multi-Plattform-Applikationen massiv. Dieser Programmierstil orientiert sich am Lean-Programming-Principle[16] und gibt dem Kunden eine grössere Wertschöpfung da der Time-to-Market-Value sinkt.

Business-Logic-Layer

Der Business-Logic-Layer bringt die geladenen Daten in Relation zueinander und bereitet die Datensätze auf. Der UI-Layer kann nach dem initialen Laden der Daten durch den Access-Layer in Echtzeit auf die Daten zugreifen und die Relationen zwischen den Datensätzen über Referenzen traversieren. Der UI-Layer hat auch die Möglichkeit gleiche Daten mehrmals abzufragen, denn die Daten befinden sich zur Laufzeit der Anwendung im Memory.

Data-Access-Layer

Der Data-Access-Layer ist zuständig für das Laden der Daten von den verschiedenen Endpunkten. Er erlaubt den transparenten Austausch von Endpunkten und Protokollen indem der Business-Logic-Layer die Endpunkte nicht direkt angibt, sondern über URIs die benötigten Ressourcen anfordert. Um die Offline-Verfügbarkeit zu gewährleisten, wurde ein HTTP-Handler und ein Filesystem-Handler implementiert. Da die Handler dynamisch aufgebaut wurden, kann zu jeder Zeit ein weiterer Handler, wie z.B. ein Bluetooth-Handler, implementiert werden.

Die Implementation des Handlers ist in zwei Klassen unterteilt:

Factory	Request-unabhängige Funktionalitäten
Client	Request-spezifische Funktionalitäten

Die *Factory* generiert neue *Clients* und injected alle geteilten Ressourcen in den Client. Der *HTTPClient* von .NET kann dadurch über verschiedenen Implementationen verteilt werden.

Der *Client* implementiert das Senden und Empfangen eines Requests und enthält alle Request-spezifischen Daten, wie z.B. die HTTP-URL des Endpunktes. Diese Klasse wird für jeden Request neu erstellt, um Thread-Safety zu gewährleisten.

Der Business-Layer nutzt einen *RequestBuilder*, um ein *Request*-Object zu generieren, welches die Anfrage an den Access-Layer darstellt. Dieser Builder erlaubt die Generierung von URN[17], um die Endpunkte eindeutig zu identifizieren. Das Ziel von URN ist, dass die Identifikation der Endpunkte in der Business-Logic unabhängig von der verwendeten Kommunikationstechnologie ist.

Der Builder trennt den Code des Business-Layers vom Code des Access-Layers.

Die Request-Klasse ist ein kleiner Wrapper, der sich für alle möglichen Verbindungsparameter als Tuple verhält.

5.3.2 Business-Model

Aus dem Anhang F kann das *Business-Layer-Model* entnommen werden. Das Modell zeigt die Übertragung der Objekte in der echten Welt auf. Die Aggregatsklasse ist *Instance*, von welcher aus alle Relationen erstellt werden.

Instance & Instance-Part	Darstellung Maschine & Komponentenbaum
ServiceCase & Technician	Metadaten der Service Abarbeitung
Activity & Modification	Änderungen an der Maschine
InstanceProperty & IPValue	Eigenschaften der Maschine, OS Version etc.
SensorData	Aktuelle Werte und MetaDaten der Sensoren

TABELLE 5.9: Relationen der Aggregatsklasse

Die Referenzen erlauben dem UI-Layer die zusammenhängenden Informationen abzufragen. Bei dem Design der Applikation wurde viel Wert darauf gelegt, dass der UI-Layer und das Domain-Model eng aufeinander abgestimmt sind.

Das Aggregat ist dafür zuständig den Graph im Memory der Hololens aufzubauen. Dies geschieht in folgenden Schritten:

1. Separater Request für Instance und Service Cases
2. Parsen von JSON-Antwort in Klassenbruchteile
3. Aggregat Instance wird aufgerufen
4. Instance fügt alle Service Cases in den Tree
5. Modifications werden mit den Instance-Parts verbunden
6. UI erhält die aggregierten Daten

Die Implementation des Business-Layers ist nur lesbar und wird bei Änderungsanfragen an das Backend automatisch mit den Änderungen des Backends aktualisiert.

5.4 CAD-Daten Visitor

Nachdem der Access-Layer das CAD-File geladen hat, muss dieses geparsed werden. Dazu wurde das Add-on *runtime obj importer*[18] verwendet. Das verarbeitete File wird in einer flachen Hierarchie zurückgegeben. Die Anordnung entspricht nicht den Anforderungen, da die Bauteile in einer hierarchischen Struktur benötigt werden. Damit diese Anforderung umgesetzt werden kann, muss Anhand der Baumstruktur der Instanz und seinen Parts ein Mapping zwischen dem CAD-File und der Baumstruktur geschehen. Für diese Umsetzung wurde das Visitor-Pattern[19] eingesetzt.

Folgende zwei Visitors wurden verwendet, um die erwünschte Struktur zu erhalten:

GameObjectSetupVisitor	Verantwortlich für den Aufbau der Struktur
BoxColliderResizeVisitor	Verantwortlich für das Updaten der Box Collider

Die Box-Colliders[20] beschreiben das Verhalten bei einer Kollision zwischen dem Gaze-Cursor und dem Game-Object.

5.4.1 Repository

Das Repository-Interface ist die Schnittstelle zwischen dem UI-Layer und dem Backend. Das UI kann über diese API Backenddaten anfordern und wird durch die Observables benachrichtigt, sobald die Daten angekommen sind.

Die Schnittstelle wird vom Model-Manager implementiert. Diese Klasse ist dafür zuständig die Anfragen an die betreffenden Backendservices weiterzuleiten.

5.4.2 Bounded-Contexts

Bounded-Contexts[21] ist ein Domain-Driven Design-Pattern, welches erlaubt das Domain-Model in verschiedene, eigenständige Kontexte zu unterteilen. Diese Kontexte können selbständig geladen und verwaltet werden.

Instance	Die Instanz und der Komponentenbaum
Service Case	Die Service Cases und Modifications
Sensordaten	Die Sensordaten, der jeweiligen Instanz
CAD	Die CAD-Files sind konzeptionell unabhängig von den Daten

Die Bounded-Contexts werden dazu verwendet, um die Komplexität des Servers auf verschiedene Endpunkte zu verteilen und so die Ausbaufähigkeit der Serversoftware in Zukunft zu gewähren. Diese Endpunkte sind voneinander unabhängig und können skaliert werden.

5.5 Anpassungen am Web-API

Für die Hololens-Applikation wurde die Serversoftware angepasst. Die Applikation hat andere Anforderungen und muss aus diesem Grund ein für diese Arbeit optimiertes Interface besitzen.

Die Serversoftware führt die verschiedenen Datensätze der Backing-Services [22] zusammen. Die Software wurde dem Backend-for-Frontend-Pattern [23] nachempfunden.

Die Software ist in ein Backend für die Website und eines für die Hololens unterteilt. Diese Unterteilung erlaubt die Optimierung des Backends für das Frontend. Ausserdem erlaubt das Pattern die Autonomie der Applikationen zu wahren.

Unter 5.12 können die genutzten Endpunkte und ihr Ergebnis entnommen werden.

Methode	Endpoint	Return-Type
GET	/api/hololens/instances	InstanceDto[]
GET	/api/hololens/instances/{id}	InstanceDto
GET	/api/hololens/instances/{id}/cad	302 -> CAD-File
GET	/api/hololens/instances/{id}/servicecases	ServiceCaseDto[]
GET	/api/hololens/instances/{id}/parts/{id}/ sensordata	SensorDataDto
POST	/api/hololens/servicecases/{id}/close	200 OK
POST	/api/hololens/servicecases/{id}/modify/ {type}/{partId}	ActivityDto

TABELLE 5.12: Endpunkte der Serverapplikation für die Hololens

Die Serversoftware wurde bewusst für folgende Anforderungen entwickelt:

Langsame Internetverbindungen

Minderwertige Internetverbindungen zeichnen sich durch eine hohe Paketumlaufzeit und viele Datenfehler aus.

Für die Übertragung des Datenmodells muss die Applikation nur wenige Endpunkte ansprechen. Diese Endpunkte liefern alle Daten aus den verschiedenen Services.

Diese Vorgehensweise nutzt die Handshakes der TCP Verbindung optimal. Ausserdem können bei grösseren Antworten die Fast-Resend-Eigenschaften von TCP sehr gut ausgenutzt werden.

Restful HTTP

Die Namensgebung bei den Pfaden zu den Serverressourcen wurde bewusst nach einem lesbaren Format designed. Ausserdem wurde dabei auf die Erweiterbarkeit geachtet.

Alle Endpunkte, die eine Liste von Ressourcen liefern, werden im Plural angesprochen.

Die Endpunkte, die eine spezifische Id laden sollten, werden mit */collectionname/id* angegeben. Diese Angabe sollte die Programmierer daran erinnern, dass der Endpunkt das Element in der *collection* mit *id* herausucht.

Die Methodennamen sind den CRUD-Aktionen nachempfunden. Beispielsweise bedeutet *GET*, dass Daten gelesen werden. Die *POST* Methode schreibt eine Änderung auf den Server.

Sicherheit

Der AAA-Code wurde von der bestehenden Anwendung übernommen.

Datenreferenzierung

Damit die Hololens-Applikation bestimmte Elemente in den unstrukturierten Datensätzen finden kann, sind bestimmte Konventionen einzuhalten:

Instanzdatenbaum	Das BOM-Modell der laufenden Instanz wird im Instanzdatenbaum gespeichert. Die Identifikation der Elemente des Instanzdatenbaums ergeben die Namensgebung der Elemente im CAD-File. Ausserdem muss die Identifikation des BOM-Parts deren der Sensoren entsprechen. Damit kann die Anwendung alle Sensoren eines Parts automatisch erkennen und einbinden.
Sensordaten	Die Sensordaten dürfen nur eine einzige Messreihe enthalten. Falls ein Part mehrere Messreihen aufnimmt wird auf dem Server eine Fehlermeldung generiert und die Verarbeitung abgebrochen.

Kapitel 6

Learnings

Dieses Kapitel beschreibt die Erkenntnisse, welche innerhalb der Arbeit gesammelt wurden. Zuerst wird der Projektverlauf in der Retrospektive beschrieben, um aufzuzeigen in welcher Form gearbeitet wurde, welche Schwierigkeiten auftraten und in welche Bereiche am meisten Zeit investiert wurde.

Später werden die Learnings zu den drei wichtigsten Technologien Unity, MRTK und UWP genauer erläutert. Diese sind für Nachfolgearbeiten relevant und sollen dabei helfen, schneller in die Materie zu finden.

6.1 Projektverlauf in der Retrospektive

Da sich das Entwicklungsteam aus drei Entwickler zusammensetzte und die Limitation durch die Hardware bestand (lediglich eine Hololens zur Verfügung), wurde die Arbeit in die drei Teilbereiche *Prototyping und Tooling*, *Backend* und *Frontend* aufgeteilt. Nachfolgend werden diese drei Bereiche genauer beschrieben.

6.1.1 Teilbereich: Prototyping und Tooling

Da im Team noch keine Kenntnisse über die Entwicklung von Mixed-Reality-Anwendungen vorhanden waren, wurde in der Konzeptionsphase grossen Wert auf das Kennenlernen der verfügbaren Werkzeuge gelegt. Hierbei wurde anhand von Tutorials das Software Development Kit erprobt. Vor allem das Wissen zu Unity als Visualisierungstool mit dem MRTK zur Realisierung von Mixed-Reality-Plattformen musste von Grund auf erlangt werden.

Schwierigkeiten machte primär der Umstand, dass die meisten Tutorials veraltet waren, da das MRTK in einer Phase der Neuimplementation stand, um den Ansprüchen an die Hololens 2 Projekte gerecht zu werden. Da das MRTK für die Verwendung von Hololens 1 und Hololens 2 Applikationen ausgelegt ist, würde in Zukunft ein Upgrade auf die Hololens 2 vereinfacht. Wie ein solcher Upgrade gewährleistet werden kann ist im Kapitel 8 genauer beschrieben.

Wie bereits erwähnt mussten viele Implementationsfragen mittels Trial-and-Error beantwortet werden, weshalb parallel zum eigentlichen Development-Branch einen Prototype-Branch eingeführt wurde. Der Prototype-Branch diente als ungepflegte Spielwiese und beinhaltete experimentell erarbeitete Scripts, welche im Verlaufe der Zeit in den Development-Branch eingepflegt wurden. So konnten sehr zügig Lösungsansätze ermittelt werden, ohne die Codequalität zu verschlechtern.

6.1.2 Teilbereich: Backend

Der Teilbereich Backend beinhaltete alle Arbeiten, welche die Schnittstelle zwischen Server und Hololens gewährleisten. Diese umfassten vor allem die Bereitstellung, Übertragung und Aufbereitung der Daten. Dabei wurde in dem bestehenden Backend und Backbone Massnahmen ergriffen, um den Datenstand an die Bedürfnisse der Hololens anzupassen. Desweiteren mussten Endpoints eingeführt werden, welche von der Hololens-Applikation angesprochen werden konnten. Die Persistierung der Daten auf der Hololens wurde ebenfalls in diesem Zusammenhang gelöst.

6.1.3 Teilbereich: User Interface

Die Arbeiten für das User Interface beinhalteten die Visualisierung der Daten, die Benutzerführung und das Interaktionsmanagement. Grundlage für diese Arbeiten waren die Erkenntnisse aus dem Prototyping und Tooling. Zudem halfen die Beispielszenen aus dem MRTK bei Visualisierungskonzepten.

Für das User Interface wurde in dieser Arbeit am meisten Zeit investiert, da das Ausarbeiten von Visualisierungskonzepten für holographische Darstellungen eine völlig neue Komplexität mit sich brachten.

Besonders schwierig war, unter Berücksichtigung des limitierten Blickfeldes (30°) die Daten leserlich darzustellen und die Sicht auf die physische Maschine nicht zu stark zu beeinträchtigen. Ein wichtiger Faktor waren auch die Lichtverhältnisse im Raum, welche die Lesbarkeit von Daten teils stark beeinträchtigen konnten.

Umgangen wurden diese Probleme vor allem mit gleitenden Dialogen und fixierten Tooltips. Der Text wurde dabei sehr kurz gehalten, um das Sichtfeld möglichst freizuhalten.

6.2 Unity

6.2.1 Abhängigkeiten der Applikation

Abhängigkeiten zu anderen Bibliotheken werden nicht durch NuGet[24] verwaltet, denn Unity hat dafür ein eigenes System. Unter dem Verzeichnis *Assets/Plugins* können Assemblies und Projektverzeichnisse abgelegt werden, die Unity in den Editor laden sollte. Diese Dependencies sind dann auch für die Applikation verfügbar und können wie gewohnt im Visual Studio referenziert werden. Der Unity Store legt die gekauften Bundles auch in diesem Verzeichnis ab.

6.2.2 Asset-Bundles

Ein Asset-Bundle ist ein Unity-Container, welcher optimierte Daten für die Anzeige von UI-Elementen beinhaltet und gilt als Best-Practice-Ansatz. Dieser Container wird durch Unity vor der Ausführung in die native Darstellung kompiliert.

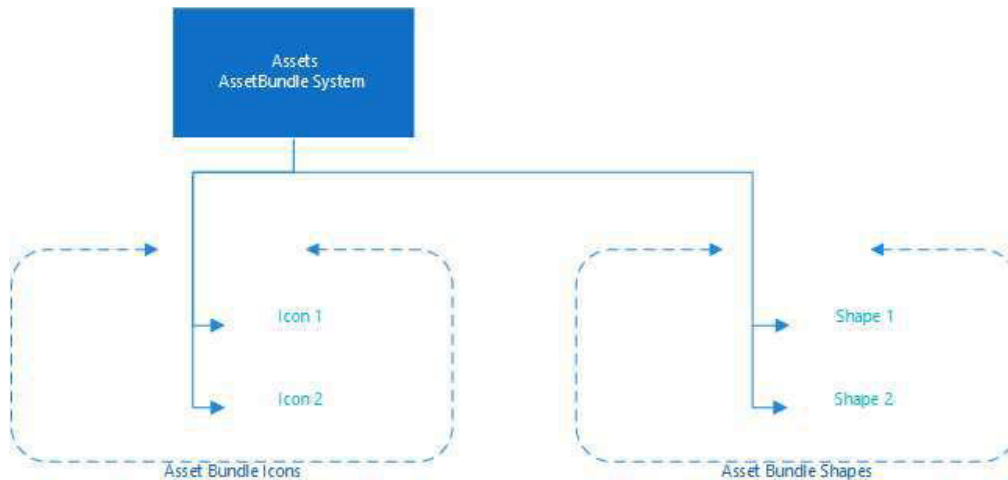


ABBILDUNG 6.1: Schematische Darstellung Asset-Bundles

Objekte können durch Angabe des Filenamens des Asset-Bundles und den Objekt-namen in Unity referenziert werden. Diese werden als native Objekte geladen und von Unity in den richtigen Typ konvertiert.

Die Asset-Bundles werden nicht standardmässig durch Unity kompiliert. Dies muss durch eine Anpassung der Build-Pipeline in einem Editor-Skript angepasst werden.

```

public class AssetBundlerBuilder: Editor {
    [MenuItem("Assets/Build AssetBundles")]
    static void BuildAllAssetBundles() {
        string assetBundleDirectory = "Assets/StreamingAssets";

        if (!Directory.Exists(assetBundleDirectory)) {
            Directory.CreateDirectory(assetBundleDirectory);
        }

        BuildPipeline.BuildAssetBundles(assetBundleDirectory,
            BuildAssetBundleOptions.None, BuildTarget.WSAPlayer);
    }
}
  
```

ABBILDUNG 6.2: Anpassung der Build-Pipeline

Durch Editor-Skripts können alle Editor-Funktionalitäten angepasst werden. Vorteilhaft ist die Konfigurierbarkeit des Editors. Dabei besteht jedoch das Risiko, dass der Editor teilweise abstürzen kann. Deshalb müssen diese Skripts mit besonderer Vorsicht geschrieben werden.

6.2.3 Prefabs

Previously fabricated gameobjects oder auch kurz Prefabs erlauben das mehrfache Laden von Spielkomponenten im Unity Editor. Beispielsweise kann ein Prefab von einem Auto erstellt werden, welches immer und immer wieder kopiert werden kann.

In der Applikation werden diese Prefabs benutzt, um Benutzerelemente zu erstellen, welche mehrfach benötigt werden. Diese werden dann zur Laufzeit geladen und dynamisch konfiguriert. In der vorliegenden Applikation sind das zum Beispiel die einzelnen Parts der Maschine.

6.2.4 Unity-Web-Request

Die Web-Request-Klasse ist eine One-Stop-Lösung für das HTTP-Protokoll. Zusammen mit der JSON-Utilities-Klasse, welche für das Parsen der JSON-Response zuständig ist, sind einfache HTTP-Zugriffe sehr schnell geschrieben. Die Klasse besitzt jedoch ein komplexes Bedienungsverhalten. Da die Klasse überprüft, ob der GUI-Thread sie aufruft, wurde sie im Projekt nicht verwendet. Stattdessen dient ein Thread-Pool dazu, die Datenoperationen auszuführen.

Während der Implementationsphase hat sich gezeigt, dass die Web-Request-Klasse inkompatibel zur Lösung mit dem Thread-Pool ist. Aus diesem Grund wurde im Access-Layer eine eigene Implementation eines High-Level-Web-Clients erstellt. Dieser Client übernimmt das Mapping von Informationen im Business-Logic-Layer auf den .NET-HTTP-Client. Für mehr Informationen über Threading siehe Sektion 6.2.5

6.2.5 Threading

Multi-Threading wird von Unity nativ nicht supported, stattdessen werden Coroutinen angeboten. [25] Eine Coroutine ist eine Parallelisierung innerhalb eines Threads. In Unity laufen diese im UI-Thread.

Wenn das Laden der Daten innerhalb des UI-Threads geschieht, speziell mit der Coroutine-Technologie, kann das UI blockiert werden. Aus diesem Grund wird das Lesen und der Aufbau der Daten in einem .NET Thread-Pool verrichtet.

Um in Unity `async/await` effizient anzuwenden, sollte nicht die .NET Implementation verwendet werden. [26] Die .NET-Implementation erstellt komplexe Stackframes, welche möglichst verhindert werden sollten. Es muss bedacht werden, dass die Applikation zu C++ transpilliert wird und damit andere Anforderungen als .NET-Code aufweisen muss.

Als Alternative für Unity wurde entschieden UniRx zu verwenden. [27] Wenn die Antwort des Servers ankommt, werden die vorherig definierten Arbeitsschritte ausgeführt und zum Ende das UI mit den neuen Daten aktualisiert.

6.2.6 JSON-Utilities

Die Klasse JSON-Utilities ist für das Parsen von JSONs zuständig. [28] Sie hat ein sehr gutes Interface zum Clientcode. Die Klasse kann jedoch keine Top-Level-Arrays

parsen. Aus diesem Grund wurde die Klasse nicht verwendet und stattdessen auf JSON.NET gesetzt. [29]

Diese Bibliothek bietet die folgenden Features:

- Übersetzen von JSON-Objekte in .NET-Objekte
- Übersetzen von JSON-Arrays in .NET-Listen
- Konfigurierbarkeit auf der Klasse

6.2.7 Logging

Es wurden verschiedene Ansätze evaluiert und es stellte sich heraus, dass das Logging unter Unity in Kombination mit Hololens sehr schwierig ist.

- Unity internes Logging
- Externes Logging in File
- Logging auf Webserver

Das Unity interne Logging funktioniert gut über die Klasse Debug. Risiken dieser Lösung bestehen darin, dass bei einer grösseren Applikation mit vielen Loggingaufrufen die benötigte Information im GUI von Unity nicht gefunden werden können. Ausserdem kann die Lösung nicht zwischen Schweregraden von Ereignissen unterscheiden. Dieser Ansatz wurde nur in der ersten Phase der Entwicklung verwendet, um eine schnelle Rückmeldung zu bekommen.

Als Erweiterung zum internen Logging wurde das externe Logging in einem File umgesetzt. Dafür wurde die Bibliothek NLog[30] verwendet. Dieses File ist mit einem Log-Viewer durchsuchbar und erlaubt ein schnelles Eingrenzen der Problematiken. Nachteil dieser Lösung ist, dass die Logging-Bibliothek schwierig in Unity zu integrieren ist. Beispielsweise sollte NLog über ein Config-File konfiguriert werden, dies ist unter der C++ Umgebung auf der Hololens jedoch nur schwer umsetzbar.

Aufgrund der Ergebnisse des internen und externen Loggings, sollte in einem weiteren Schritt das Logging über ein Web-GUI realisiert werden. Da es dafür jedoch keine Standard-Komponenten gab, wurde dieser Ansatz nicht weiter verfolgt.

6.2.8 Zenject

Unity verfügt über keine Implementation für Dependency Injection, daher wurde die Bibliothek Zenject verwendet.

Der Einsatz von Zenject kann aus folgenden Gründen gerechtfertigt werden:

- Instanziierung von Objekten, die nicht Teil der Klasse sind
- Abhängigkeitsauflösung und Validation durch Zenject
- Durch Interfaces sind alle Klassen lose Komponenten, welche jederzeit ohne Codeänderung austauschbar sind
- Trennung von Konfiguration und Klassen

- Schreiben von Test-Code ist stark vereinfacht

6.2.9 Testing

Bei der Evaluation des Test-Frameworks von Unity war dieses nicht mit der Visual Studio IDE kompatibel. Dies zeigte sich durch Fehlermeldungen im Programm, welche im Compiler nicht auftraten. Dadurch kann die IDE beim Schreiben des Codes nicht unterstützen.

Unit Tests sind auf der WSA-Plattform sehr limitiert und können nur im Editor ausgeführt werden. [31]

Deshalb wurde aufgrund begrenzter Zeit und dem Fokus auf Learnings und Ergebnisse entschieden, keine Unit Tests für den Prototypen zu schreiben.

6.3 Mixed-Reality-Toolkit

Das Mixed-Reality-Toolkit ist ein Abstraktionslayer für die plattformspezifischen Funktionalitäten der Hololens. Das Toolkit erlaubt die Konfiguration von Einstellungen der Hololens und des Verhaltens im Unity-Editor.

Das MRTK hat den Developmentprozess beschleunigt und einige Funktionalitäten zur Verfügung gestellt. Interaktionen mit den Game-Objects wurden durch ein sogenanntes *Interactable*-Script ermöglicht. Das Docking von Tooltips und Buttons erfolgte über die *Solver*- und *Bounding Box*-Scripte. So konnte vor allem auf das Design und die für diesen Use Case relevanten Implementierungen fokussiert werden.

Zum Zeitpunkt der Arbeit wurde von Microsoft aktiv weiterentwickelt. Über das Git-Repository des MRTKs war es möglich, den Entwicklungsprozess der Library nachzuverfolgen. Durch das Erstellen von Bug-Request konnte auf verschiedene Missstände hingewiesen werden, welche dann im Verlaufe der Arbeit von einem Team gereviewed und verbessert wurden. Im Rahmen dieser Arbeit wurden mehrere Fehler gefunden, welche bereits als Bugreport erfasst waren, sowie neue Fehler, welche gemeldet wurden.

Nachfolgend eine Auflistung der wesentlichen Fehler:

- Interactable Voice Command feature is confusing [32]
- Hololens Emulation: Cursor stops working after first air-tap on Hololens (Gen 1) using RC1 [33]
- Holographic Remoting: Cursor stops working after first air-tap on Hololens (Gen 1) using RC1 [34]
- In-Editor Input Simulation not working [35]

Grundsätzlich wurde sehr schnell auf die Fehlermeldung reagiert und in manchen Fällen sogar direkt ein Bugfix realisiert. Leider konnten gewisse Bugfixes nur per manuellem Einpflegen in das Mixed-Reality-Toolkit integriert werden.

Da das MRTK aber neu entwickelt wurde und nur als Release Candidate zur Verfügung stand, gab es auch einige Restriktionen. Zum Beispiel waren die meistens Tools nur für die statische Konfiguration über den Editor ausgelegt. Dies zeigte sich vor

allem dadurch, dass Classmembers der einzelnen Instanzen während der Laufzeit *private* waren und von aussen nicht manipuliert werden konnten. Dies verunmöglichte eine vollumfängliche Dynamik des Programms. Daher musste an gewissen Stellen das MRTK manuell angepasst werden, um die Anforderungen zu erfüllen.

Nachfolgend eine Auflistung der manuellen Anpassungen:

- InteractableMaterialTheme.cs - Zeile 48: *Host.SetActive(property.Values[index].Bool);* entfernt
- Interactable.cs - *Awake()* von *private* auf *public* geändert, da kein Setter existierte
- ToolTipSpawner.cs - *toolTipText* von *private* auf *public* geändert, da kein Setter existierte
- ToolTip.cs - Zeile 457 *background.OnContentChange(localContentSize, LocalContentOffset, contentParent.transform);* auskommentiert
- AppBar.cs - Indexes der Buttons geändert und *DisplayOrder* (Zeile 26 und 490)

Zusammen mit der Community konnte schliesslich auch ein brauchbarer Workaround ausgearbeitet werden, um die Dynamik in der Applikation trotzdem zu gewährleisten. Dieser beinhaltet das Erstellen eines Prefabs, welches bereits die Tools des MRTKs zugewiesen und vorkonfiguriert hat. Das Prefab konnte dann während der Laufzeit über den Asset-Handler von Unity nachgeladen und an der richtigen Position angezeigt werden.

Es kann davon ausgegangen werden, dass diese Restriktionen im Verlaufe der Zeit im MRTK verbessert werden, so dass zukünftig die Funktionalitäten vollständig zur Laufzeit konfiguriert werden können.

6.4 Universal Windows Platform

6.4.1 Universal Windows Platform

Die Universal Windows Platform ist eine Programmierumgebung um native Applikationen zu entwickeln. Diese laufen unabhängig von der Windows-Edition oder Hardware auf allen Windows-Systemen ab Version 10. [36] Auch Augmented-Reality-Applikationen für die Hololens werden auf dieser Plattform betrieben.

Diese neuartige Programmierumgebung bietet dem Nutzer Sicherheit, Einfachheit in der Wartung und universelle Übertragbarkeit. Die Applikationen können vom Microsoft Store heruntergeladen werden. [37]

Die Bibliotheken der Universal Windows Platform sind nicht Teil von Unity. Unity ist eine C++-Applikation, die eine Mono-Laufzeit eingebaut hat für die Ausführung von C#-Code. [38]

Im Editor wird der C#-Code in IL kompiliert und unter einer plattformunabhängigen Version von der Mono-Laufzeit ausgeführt. Die Laufzeitbibliotheken von Mono und Unity sind in dieser Laufzeit verfügbar.

Die UWP nutzt einen speziellen Compiler für den Compilegang. Der IL2CPP-Compiler (siehe 6.4.2) macht die Universal Windows Platform Bibliotheken (UWP API) zum Aufrufen im Spielecode verfügbar.

Um das Programm auch im Editor testbar zu machen, muss für jede Plattform ein bedingter Compilegang ausgeführt werden. Die Klassen, die das UWP API aufrufen, müssen ihre Implementation bei der Ausführung im Editor verstecken, um Errors auf Grund fehlender API zu verhindern.

6.4.2 IL2CPP-Compile-Prozess

Unity verwendet für Windows-Store-Applikationen eine native Implementation des IL2CPP-Compilers. [39] Der benötigte IL2CPP-Prozess wird von Microsoft empfohlen [40], um die Framerate auf einem hohen Niveau zu halten und somit flüssige Applikationen zu ermöglichen. Denn die Hololens bietet mit ihrer Hardware nicht die gleiche Leistung wie ein handelsüblicher Computer.

Nachfolgend sind alle notwendigen Compileschritte aufgezeigt:

1. Öffnen des Projektes in Unity Editor
2. Kompilierung des Codes in .NET-Compiler
3. Kompilierung der Assets in Asset-Bundles
4. Kompilierung des Codes in .NET-Compiler-Release-Mode
5. Kompilierung der Shaders
6. Linken der Plugins und Entfernen von ungenutztem Code
7. Kompilierung des .NET-IL-Codes in C++
8. Kompilierung des C++-Codes als nativer Code
9. Linken des nativen Codes
10. Übertragung des Programms auf die Hololens
11. Ausführung der Applikation auf der Hololens

6.4.3 Build auf einem Laptop

Prozessor: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2 Kern

RAM: 8 GB

Grafikkarte: Intel HD Graphics 520

Abgesehen vom ersten Build kann mit 20 bis 30 Minuten gerechnet werden. Der Prozess nutzt den Computer komplett aus, daher können in dieser Zeit keine anderen Arbeiten erledigt werden.

6.4.4 Build auf einer CAD Maschine

Prozessor: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz 6 Kern

RAM: 32 GB

Grafikkarte: NVIDIA GeForce GTX 1060 6 GB

Abgesehen vom ersten Build kann mit ca. 10 Minuten gerechnet werden. Der Prozess nutzt den Computer nicht komplett aus, daher können in der Zwischenzeit andere Arbeiten erledigt werden.

6.5 Voice

6.5.1 Spracherkennung

Die Spracherkennung unter Windows wird lokal auf dem Gerät bearbeitet. Der Programmierer kann über die Universal Windows Platform API auf die Speech Recognition API zugreifen .[41]

Die Spracherkennungsalgorithmen sind auf allen Windows 10 basierten Betriebssystemen vorhanden und können von Universal Windows Platform Applikationen benutzt werden. Somit kann die Applikation für die Hololens auch auf einem Windows 10 Computer ausgeführt werden und Spracherkennung anbieten .[42]

Das MRTK übernimmt die Einbindung der Speech-Recognition-API. Diese Einbindung ist nicht einfach, da es sich um eine proprietäre Windows-API handelt, die Unity nicht eingebaut hat. Für mehr Informationen bezüglich der Einbindung von UWP-API siehe Sektion 6.4.1.

Wie unter 6.3 beschrieben wurde, ist der Code in der Spracherkennung fehlerhaft. Währenddem die Windows-API korrekte Resultate liefert, kann das MRTK die Events der Spracherkennung nicht an die korrekte UI-Komponente weiterleiten. Deshalb muss das Element mit dem Gaze-Pointer angesehen werden, um die Aktion korrekt auszulösen.

Durch diese Einschränkung fühlt sich die Nutzung von Spracherkennung in einer Unity Applikation zur Zeit unnatürlich an. Es wird davon ausgegangen, dass das Issue mit der nächsten Versionen des MRTKs gelöst sein wird.

6.5.2 Sprachausgabe

Die UWP-Applikationen können Sprachausgabe mithilfe einer Text-to-Speech-Engine nutzen.

Das MRTK besitzt nativ keine Möglichkeit Sprachausgabe anzubieten. Es muss eine Helferklasse aus dem MRTK in den Applikationscode kopiert werden. Diese Helferklasse abstrahiert die Windows-API, um die Problematiken die unter 6.4.1 beschrieben werden zu vermeiden .[43]

Die Helferklasse ermöglicht es die Text-to-Speech-Funktionalität sehr einfach einzusetzen. Es muss ein Text-in-Speech Synthesis-Markup-Language (SSML) aufbereitet werden, um der Sprachausgabe Informationen über die Aussprache mitzugeben .[43]

SSML ist ein XML basiertes Format, das den auszusprechenden Text mit Informationen über spezifische Aussprachen der Passagen annotiert. Dies erlaubt dem Programm die Aussprache anzupassen, um eine natürlichere Ausgabe zu erhalten.

Die Sprachausgabe von Windows generiert eine Datei im Windows-Audio-Format. Dieses Format wird von der Helferklasse in Unity `AudioClip` transformiert. Der Klasse kann eine `AudioSource` zugewiesen werden, um den Text abzuspielen. Auf diese Weise kann die Stimme im 3D-Raum platziert werden und bewegt sich mit dem Objekt im Raum.

Kapitel 7

Ergebnisse

In diesem Kapitel wird der Stand der Applikation am Ende der Arbeit beschrieben. Es wird anhand der Anforderungen aus dem Kapitel 3 gezeigt was umgesetzt wurde. Desweiteren werden die nicht-funktionalen Anforderungen geprüft und die erzielten Resultate dokumentiert.

7.1 Erfüllte Anforderungen

Die folgende Liste beschreibt anhand der Use Cases in welchem Stadium sich die Applikation heute befindet.

UC01	Bronze	Instanz- und Partdaten werden dynamisch geladen.	Umgesetzt
UC02	Bronze	CAD-Modell der Instanz wird dynamisch geladen und visualisiert.	Umgesetzt
UC03	Bronze	Mapping zwischen Instanz- und Partdaten und CAD-Modell.	Umgesetzt
UC04	Bronze	Navigation durch den CAD-Modell-Baum.	Umgesetzt
UC05	Bronze	Teile der Visualisierung können aus- und wieder eingeblendet werden.	Umgesetzt
UC06	Silber	Visualisierung als Layer über der physischen Maschine (Spatial Mapping).	Umgesetzt
UC07	Gold	Montageanleitung für den Servicetechniker für einen spezifischen Service Case anzeigen.	Offen
UC08	Gold	AR-Brille kann ohne eine Internetverbindung verwendet werden.	Teilweise
UC09	Bronze	Instanzinformationen werden angezeigt.	Umgesetzt

UC10	Bronze	Service Cases zur Instanz werden angezeigt.	Umgesetzt
UC11	Bronze	Service Case kann geschlossen werden.	Umgesetzt
UC12	Bronze	Partinformationen werden angezeigt.	Umgesetzt
UC13	Bronze	Modifikationen zu den Parts werden angezeigt.	Umgesetzt
UC14	Bronze	Parts können gewartet werden, es wird dafür eine Modifikation erstellt werden.	Umgesetzt
UC15	Bronze	Parts können ausgetauscht werden, es wird dafür eine Modifikation erstellt werden.	Umgesetzt
UC16	Bronze	Sensordaten zu den Parts werden angezeigt.	Umgesetzt
UC17	Silber	Sensordaten werden während der Laufzeit aktualisiert	Umgesetzt
UC18	Gold	Für alle Funktionen können Voice Commands verwendet werden.	Umgesetzt
UC19	Gold	Audioausgabe bei Funktionen.	Offen
UC20	Gold	Visualisierte Elemente können manipuliert werden (vergrössern, verkleinern, rotieren, verschieben).	Umgesetzt

TABELLE 7.3: Umsetzungsstadium der Use Cases

Von 20 Use Cases konnten bis auf zwei alle umgesetzt werden, wobei beim UC08 nur Read-Only-Funktionen offline verfügbar sind. Für die Komplexität verschiedener Zeitlinien bei Write-Funktionen fehlte sowohl die Zeit als auch der Fokus.

7.2 Nicht-funktionale Anforderungen

Diese Sektion beschreibt die erreichten nicht-funktionalen Anforderungen und wie diese erzielt wurden.

Detaillierte Informationen über die Anforderungen an den Messpunkt können unter 3.2 entnommen werden.

7.2.1 Funktionalität

Genauigkeit

Eigenschaft	Keine Relation zu Maschine	Ungefähre Genauigkeit	Genaue Übereinstimmung
Modellposition		✓	✓
Sensordaten		✓	
Bedienungselemente			✓

TABELLE 7.4: Landing Zones Genauigkeit

Das Image Target (Bild welches durch Vuforia erkannt wird) ist ausschlaggebend für die Genauigkeit der Positionierung. Da man das Image Target frei an der Maschine positionieren kann, kann das Modell manuell justiert werden.

Die Sensordaten befinden sich ca. 1.5 Meter über den dazugehörenden Parts. Sie sind damit abhängig von der Position des Parts. Das selbe Verhalten wurde bei den Bedienungselementen angewandt.

7.2.2 Zuverlässigkeit

Fehlertoleranz

Eigenschaft	Keine Toleranz	Ignorieren	Automatische Behandlung
Daten lesen		✓	✓
Daten schreiben		✓	
Fehlerhaftes CAD		✓	

TABELLE 7.5: Landing Zones Fehlertoleranz

Wenn die Maschinendaten einmal heruntergeladen wurden, sind sie offline verfügbar und können somit fehlerhafte Lesevorgänge des Servers überbrücken.

Beim Schreiben wird ein fehlerhaftes Schreiben ignoriert. Dem Nutzer wird ein dies aktuell nicht mitgeteilt. Dies bietet eine Verbesserungsmöglichkeit für spätere Arbeiten.

Fehlerhafte CAD Daten führen zu einer leeren Applikation ohne ein Modell. Die Applikation stürzt zwar nicht ab, es muss aber manuell neu gestartet werden. Auch hier gibt es Verbesserungspotential für fortführende Arbeiten.

7.2.3 Benutzbarkeit

Verständlichkeit und Erlernbarkeit

Eigenschaft	Unverständlich	Verständlich nach Übung	Natürlich, Offensichtlich
Menüstruktur			✓
Service Cases		✓	
Manipulationen		✓	

TABELLE 7.6: Landing Zones Verständlichkeit und Erlernbarkeit

Gemäss unseren User-Acceptance-Tests ist die Menüstruktur selbsterklärend. Dies wurde durch gute Piktogramme und einer einfachen Navigation erreicht.

Die Menüführung der Service Cases ist mit einer kurzen Einführung gut bedienbar.

Die Manipulationen erfordern etwas Übung, die Handgriffe müssen sehr genau ausgeführt werden.

Bedienbarkeit

Eigenschaft	Unbedienbar	Bedienbar nach Übung	Trifft bei jeder Bedienung
Menüstruktur			✓
Service Cases			✓
Manipulationen		✓	

TABELLE 7.7: Landing Zones Bedienbarkeit

Die User-Acceptance-Tests haben ergeben, dass die Bedienbarkeit der Menüstruktur gut ist.

Die Menüs der Service Cases sind klar ersichtlich für die Testpersonen.

Die Testpersonen hatten Probleme mit der räumlichen Darstellung, da dies bei der ersten Verwendung von der Hololoens noch sehr ungewohnt ist.

7.2.4 Effizienz

Fliessend

Eigenschaft	< 20 FPS	20 - 50 FPS	> 50 FPS
Programm			✓
Sensor Daten			✓
Manipulationen			✓

TABELLE 7.8: Landing Zones Bildrate

7.2.5 Übertragbarkeit

Anpassbarkeit

Applikation	Unveränderbar	Abänderbar	Starke Ausbaubarkeit
Server	✓		
Hololens			✓
Dashboard		✓	
User Interface			✓

TABELLE 7.9: Landing Zones Anpassbarkeit Applikationen

Während der Entwicklung wurde stets darauf geachtet, dass Änderungen ohne grosse Auswirkungen vorgenommen werden können.

Technical Dept

Layer	Sehr viel dept	Dept vorhanden	Wenig Dept
Server	✓		
Access-Layer	✓		
Business Logic Layer			✓
User Interface			✓

TABELLE 7.10: Landing Zones Technical Dept Layers

Die Serversoftware sollte in einer späteren Arbeit überarbeitet werden. Eine genaue Beschreibung der Schwächen des serverseitigen Codes ist unter Sektion 8.2.3 aufgeführt.

Der Access-Layer wurde von einem anderen Projekt übernommen und sollte auf die Anforderungen der Hololens-Applikation angepasst werden.

Die anderen Layers wurden spezifisch für die Applikation geschrieben und haben somit wenig technical dept.

Installierbarkeit

Applikation	Komplizierte Installation	In-	Möglich mit Anleitung	Intuitiv
Server			✓	
Hololens	✓			
Dashboard			✓	

TABELLE 7.11: Landing Zones Installierbarkeit

Die Server Applikation und das Dashboard können mit einer Installationsanleitung gut installiert werden.

Die Installation der Hololens-Applikation bedarf Expertenwissen. Sie könnte über den Microsoft Store verteilt werden, um die Installation zu vereinfachen. Aus den in Sektion 4.1.7 beschriebenen Gründen wurde auf ein solches Deployment verzichtet.

7.3 Geschwindigkeit

Innerhalb der Arbeit konnte eine stabile und performante Applikation implementiert werden. Die Bildfrequenz der Applikation ist sehr wichtig, um dem Nutzer ein angenehmes Gefühl während der Bedienung mit der Hololens zu gewährleisten. Bei einer zu niedrigen Bildfrequenz kann es zu Unannehmlichkeiten kommen, die sich mit Schwindel und/oder Übelkeit zeigen.

Unity durchläuft in sekundenschnelle mehrere Iterationen, in welchen das Bild aufgebaut und die Hologramme gezeichnet werden. Die Anzahl der Aktualisierungen pro Sekunde werden in der Masseinheit *Frames Per Second* (FPS) gemessen.

Folgend werden die erreichten Werte aufgelistet:

Ausführungsmodus	durchschnittliche FPS
Unity Emulator	60 FPS
Holographic Remoting	60 FPS
Hololens Emulator Debug Build	40 FPS
Hololens Emulator Release Build	60 FPS
Hololens Debug Build	40 FPS
Hololens Release Build	60 FPS

TABELLE 7.12: Erreichte Geschwindigkeitswerte

Der IL2CPP Prozess kompiliert den C#-Code von Unity in C++, um damit eine native Applikation zu bauen, die auf dem Prozessor der Hololens ausgeführt werden

kann. Dieser Prozess beschleunigt die Aktualisierungen pro Sekunde und schafft damit auch bei der vorliegenden Applikation die notwendige Geschwindigkeit von 60 FPS.

7.4 Dynamik

Als Vorlage zu dieser Arbeit diente eine an der HSR entwickelte Arbeit [44]. Diese beinhaltet eine Maschine, die statisch in den Szenenkontext geladen und vorkonfiguriert wurde. Die gesamte Funktionalität ist demnach über den Editor definiert und kann während der Laufzeit nicht angepasst werden, was das Nachladen von Maschinendaten verunmöglicht.

In dieser Arbeit wurde das Hauptaugenmerk auf die volle Dynamik der Applikation gesetzt. Alle Daten werden in Echtzeit über Api heruntergeladen und interpretiert. Der Szenenkontext wird anhand dieser Daten aufgebaut und visualisiert. Entsprechend kann jede Art von Produktinstanz mit dem zugehörigen BOM und den Sensordaten in der Applikation angezeigt werden. Die einzige Voraussetzung ist ein BOM und ein passendes CAD-File, welches als .obj-File auf dem Server platziert wurde.

7.5 Optimierung des Speicherplatzes

Die Serverapplikation und die UWP-Applikation für die Hololens sind so implementiert, dass diese geringstmöglichen Festplattenspeicher nutzen.

Serverseitig werden die CAD-Daten für eine Instanz und den jeweiligen Bauteile mittels Seriennummer ermittelt. Das heisst, dass alle Maschinen eines Typs auf das gleiche File zeigen und somit nicht für jede Instanz ein eigenes CAD-File gezeichnet werden muss. Im Prinzip genügt es einen Maschinentyp in seinem kompletten Detailgrad zu zeichnen und als CAD-File auf dem Server zu hinterlegen. Über die BOM-Struktur wird dann für jede Instanz das passende File geladen und die korrekten Bauteile im CAD-File ermittelt, welche dann als Game-Objects in die Szene geladen werden.

Neben dem Speicherplatz ist ein weiterer Vorteil dieser Lösung, dass das DRY-Prinzip nicht verletzt wird. Wenn ein Maschinentyp neue Module erhält, müssen diese nur in einem CAD-File aktualisiert werden. Den Rest erledigt die Applikation durch das intelligente Mapping.

Seitens Hololens wurde darauf geachtet, dass die Ressourcen, wie Materials, Prefabs, Icons, usw. als sogenannte Asset-Bundles gebündelt werden. Dies optimiert sowohl Festplattenspeicher, wie auch die Ladegeschwindigkeit der Applikation.

7.6 Ladegeschwindigkeit

Die UWP-Applikation lädt alle Daten zur aktuellen Maschine vom ERP-System während des Aufstarten der Applikation. Damit der Servicetechniker nicht viel Zeit

verliert und die Applikation skaliert, ist die Ladegeschwindigkeit bestmöglich optimiert.

Die Applikation hat in den Messungen innerhalb von 30 bis 60 Sekunden alle notwendigen Daten geladen und eingelesen. In dieser Zeit bleibt die Oberfläche weiterhin responsive, da die Arbeiten in Subthreads ausgelagert sind.

In Anbetracht des Kommunikationsaufbaus inklusive Sicherheitsaspekte, der grossen Datenmenge des CAD-Files und des Setups für die Visualisierung der Daten, ist dies ein befriedigendes Ergebnis.

7.7 Ausbaubarkeit

Die Applikation hat eine Architektur, die den Ausbau der Anwendung sehr einfach zulässt. Wie bis anhin wurde auch in dieser Ausbaustufe darauf geachtet, dass das Naming, die Struktur und der Aufbau der Domäne angepasst ist.

Kapitel 8

Schlussfolgerung und Ausblick

In diesem Kapitel werden rückblickend auf das Projekt die wichtigsten Schlussfolgerungen zusammengefasst. Der Implementationsprozess, wie auch das Resultat werden dabei kritisch durchleuchtet. Zusätzlich sind Ausblicke auf Folgearbeiten festgehalten.

8.1 Schlussfolgerung

Die prinzipielle Umsetzbarkeit und Konzeption einer AR-Anwendung für die HoloLens konnte in diesem Projekt gezeigt werden. Im Speziellen das Einlesen von CAD-Daten, Instanzdaten und die Modifikation von Service Cases wurden erfolgreich umgesetzt.

Während der Arbeit sind auch die Limiten der HoloLens 1 ersichtlich geworden. Beispielsweise erlaubt die limitierte Sichtweite von 30 ° nur eine Ansicht des Gesamtmodells aus einer Distanz von mindestens zwei Metern. Auch die Prozessorleistung kommt während dem Laden der Daten an seine Grenzen. Mit 30 bis 60 Sekunden ist die Initialisierungszeit relativ lange.

Desweiteren sind die Entwicklungswerkzeuge, besonders das MRTK, noch in einem ungenügenden Zustand und es fehlen wichtige Dokumentationen. Auch die Kompilationspipeline ist nicht optimal gelöst. Die Kompilierungszeiten sind sehr lange, was eine effiziente Entwicklung erschwert.

8.2 Ausblick

8.2.1 Portierung zu HoloLens 2

Die Applikation sollte nach einer Kompilation auch auf der HoloLens 2 laufen. [45] Dies konnte jedoch nicht abschliessend getestet werden, da ein entsprechendes Gerät nicht zur Verfügung stand.

Die Nutzerschnittstelle wurde auf die Möglichkeiten der HoloLens 1 ausgerichtet. Um die Kompatibilität zur HoloLens 1 zu ermöglichen, sind die erweiterten Features der HoloLens 2 nicht berücksichtigt. Es ist jedoch denkbar, über die Settings des MRTKs weitere Handgesten zu definieren, wie die Manipulation mit beiden Händen.

8.2.2 Ausbau der Hololens-Applikation

Die Applikation kann in einer Folgearbeit mit den im Abschnitt 8.2.1 besprochenen Änderungen ausgebaut werden.

Desweiteren könnten zusätzliche UI-Konzepte in die Arbeit integriert werden. Beispielsweise ist eine interaktive Montageanleitung mit 3D-Bildern, die dem Servicetechniker zur Verfügung gestellt wird, denkbar. Die momentane Implementation der BOM-Datenbank müsste um 3D-Skizzen der Baugruppen und Ersatzteilen erweitert werden. Mithilfe dieser Zeichnungen würde der Servicetechniker bei den Reparaturarbeiten unterstützt werden.

Weiter ist denkbar, dass die Sprachsteuerung ausgebaut wird. Ein erweitertes Backbone müsste die Speicherung von Wartungsprotokollen, Maschineneinstellungen und Kundenänderungen erlauben. Diese Daten könnten dann mit der Sprachsteuerung von der Hololens abgefragt und aktualisiert werden.

8.2.3 Aufarbeitung Server

Der Code auf dem Webserver muss substanziell aufgearbeitet werden. Im Verlaufe der Zeit wurde immer mehr Logik auf dem Server implementiert. Mittlerweile ist ein Refactoring nötig, um die Codequalität und Wartbarkeit zu verbessern.

Es wird nahegelegt, den Code nach dem heutigem Wissensstand zu überarbeiten. So müsste zum Beispiel eine einheitliche Namensgebung über alle Services erfolgen. Beispielsweise wurde für eine Instanz der Namen Machine, Instance und Product verwendet.

Ausserdem kann gewisse Logik zusammengelegt, optimiert oder entfernt werden. Momentan übernimmt das Backend viele Filterungsaufgaben, die eigentlich im Backbone erfolgen müssten. Auch im Cockpit und in der Hololens-Applikation muss deshalb unnötig gefiltert werden. Dies führt zu schlechteren Antwortzeiten für die Clients.

8.2.4 Vuforia

Die Applikation nutzt zurzeit die Beispieldatenbank von Vuforia. Innerhalb der Arbeit war es nicht möglich eine eigene Datenbank mit Targets aufzubauen. Eine Möglichkeit wäre, dass anstelle des Beispiel-Targets ein Bild des HSR-Logos genommen wird.

Eine weitere Alternative wäre, dass die CAD-Daten der Maschine in Vuforia geladen werden, um die Maschine selber zu erkennen. Dies würde die Notwendigkeit des Targets abschaffen.

8.2.5 Mixed-Reality-Toolkit

Das MRTK wurde zur Zeit des Schreibens weiterentwickelt. Bis diese Entwicklung auf einem befriedigenden Stand ist, wird es sicherlich noch einige Breaking-Changes geben. Dies bedeutet, dass bei Updates ggf. auch neuer Code geschrieben werden muss.

Es wird davon ausgegangen, dass sich die Situation bis Ende des Jahres 2019 stabilisieren und eine solide Version zur Verfügung gestellt wird. Ab diesem Zeitpunkt sollte sich die Entwicklung von Hololens-Anwendungen vereinfachen, da mit vollständiger Dokumentation und weniger Bugs im MRTK gerechnet werden kann. Zum aktuellen Zeitpunkt sind 420 offene Fehler im Repository von MRTK bekannt. Auch der Umstand, dass das MRTK zum jetzigen Zeitpunkt sehr statisch ist und wenig Dynamik zulässt, müsste sich von Version zu Version verbessern.

8.2.6 Hardware

Es wird empfohlen, dass leistungsstarke Computer für die Entwicklung von holographischen Applikationen verwendet werden. Die in dieser Arbeit verwendeten Laptops waren für einen performanten Buildprozess eher ungeeignet. Bei fortführenden Arbeiten sollten CAD-Computer zur Verfügung gestellt werden, die die Buildzeit um den Faktor 10 verbessern.

8.2.7 Wirtschaftlicher Nutzen

Durch die Umsetzung des Projekts konnte gezeigt werden, dass der gegebene Use Case in der Erweiterten Realität umgesetzt werden kann.

Für eine Firma ist die Umsetzung einer solcher Anwendung momentan noch nicht profitabel. Anhand der Vorstudie über den Return of Investment, welcher im Kapitel 4.4 behandelt wurde, wird dieser erst zu spät erreicht. Einerseits sind zurzeit die Entwicklungskosten zu hoch, andererseits unterstützt das Endprodukt den Servicetechniker noch nicht genügend.

Man kann davon ausgehen, dass die Entwicklungskosten sinken, sobald bessere Lösungsansätze vorhanden sind. Zum einen werden die Entwicklungswerkzeuge immer ausgereifter und verlangen weniger Workarounds um das Feature umzusetzen, zum anderen ändern sich die Entwicklungsparadigmen und Bibliotheken unter Unity.

Literatur

- [1] ISO/IEC JTC 1/SC 7 (März 2011). URL: <https://www.iso.org/standard/35733.html> (besucht am 10.06.2019).
- [2] (20. Juli 2011). URL: <http://wirfs-brock.com/blog/2011/07/20/introducing-landing-zones/> (besucht am 11.06.2019).
- [3] (). URL: <https://www.google.com/glass/start/> (besucht am 13.06.2019).
- [4] X Company (2019). URL: <https://www.x.company/glass>.
- [5] (21. März 2018). URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details> (besucht am 13.06.2019).
- [6] Microsoft (2019). URL: <https://www.microsoft.com/de-lu/hololens>.
- [7] Unity Technologies (2019). URL: <https://unity3d.com>.
- [8] Microsoft (2019). URL: <https://microsoft.github.io/MixedRealityToolkit-Unity/MRTK-vNext.html>.
- [9] Google Developers (2019). URL: <https://developers.google.com/ar>.
- [10] Microsoft (2018). URL: <https://docs.microsoft.com/en-us/windows/uwp/publish/app-submissions>.
- [11] Microsoft (2018). URL: <https://docs.microsoft.com/en-us/windows/uwp/publish/choose-visibility-options>.
- [12] Microsoft (2018). URL: <https://docs.microsoft.com/en-us/sccm/apps/deploy-use/manage-apps-from-the-windows-store-for-business>.
- [13] (). URL: https://en.wikipedia.org/wiki/Single_responsibility_principle.
- [14] Oliver Kopp (2019). URL: <https://github.com/adr/madr/blob/master/docs/adr/template.md>.
- [15] Microsoft (). URL: <https://visualstudio.microsoft.com/xamarin/>.
- [16] Tom Poppendieck Mary Poppendieck Thomas David Poppendieck (2003). URL: https://books.google.ch/books?id=hQk4S7asBi4C&pg=PA182&redir_esc=y.
- [17] J. Klensin P. Saint-Andre Filament (2017). URL: <https://tools.ietf.org/html/rfc8141>.
- [18] Dummiesman (). URL: <https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547>.
- [19] Ralph Johnson Erich Gamma Richard Helm (). URL: https://www.exlibris.ch/de/buecher-buch/deutschsprachige-buecher/erich-gamma/design-patterns/id/9783826697005?gclid=CjwKEAjw__fnBRCNpvH8iqy4x14SJAC4XERPSXkV_CD1At6GAyGzF9c1MJo4dBjkq952xxAe6D9DPhoCcoLw_wcB&gclsrc=aw.ds.
- [20] Microsoft (). URL: <https://docs.unity3d.com/Manual/class-BoxCollider.html>.
- [21] Martin Fowler (2014). URL: <https://martinfowler.com/bliki/BoundedContext.html>.
- [22] Adam Wiggins (2017). URL: <https://12factor.net/backing-services> (besucht am 10.06.2019).
- [23] Sam Newman (18. Nov. 2015). URL: <https://samnewman.io/patterns/architectural/bff/> (besucht am 10.06.2019).

- [24] Microsoft (2019). URL: <https://www.nuget.org/>.
- [25] Unity (2019). URL: <https://docs.unity3d.com/Manual/Coroutines.html>.
- [26] (2019). URL: <https://github.com/Cysharp/UniTask>.
- [27] (2019). URL: <https://github.com/neuecc/UniRx>.
- [28] (2019). URL: <https://json.org>.
- [29] (2019). URL: <https://www.newtonsoft.com/json>.
- [30] (2019). URL: <https://nlog-project.org/>.
- [31] Unity (2019). URL: <https://docs.unity3d.com/Manual/testing-editor-test-runner.html>.
- [32] Microsoft (2019). URL: <https://github.com/microsoft/MixedRealityToolkit-Unity/issues/4598>.
- [33] Simon Müller (2019). URL: <https://github.com/microsoft/MixedRealityToolkit-Unity/issues/4094>.
- [34] RobDesrosiers (2019). URL: <https://github.com/microsoft/MixedRealityToolkit-Unity/issues/3909>.
- [35] Simon Müller (2019). URL: <https://github.com/microsoft/MixedRealityToolkit-Unity/issues/3883>.
- [36] Microsoft (7. Mai 2018). URL: <https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide> (besucht am 10.06.2019).
- [37] Microsoft (). URL: <https://docs.microsoft.com/en-us/windows/uwp/publish/> (besucht am 10.06.2019).
- [38] (2. Dez. 2009). URL: <https://answers.unity.com/questions/8555/what-is-mono-is-it-a-compiler-a-language-or-what.html> (besucht am 11.06.2019).
- [39] Unity (2018). URL: <https://docs.unity3d.com/Manual/IL2CPP.html>.
- [40] Microsoft (2018). URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/holograms-100>.
- [41] Microsoft (). URL: <https://docs.microsoft.com/en-us/windows/uwp/design/input/speech-recognition> (besucht am 10.06.2019).
- [42] Microsoft (). URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input> (besucht am 10.06.2019).
- [43] (11. Okt. 2017). URL: https://github.com/microsoft/MixedRealityToolkit-Unity/blob/htk_development/Assets/HoloToolkit/Utilities/Scripts/TextToSpeech.cs (besucht am 11.06.2019).
- [44] Simon Hersche (2017).
- [45] (12. Juni 2019). URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtrk-porting-guide/> (besucht am 13.06.2019).