Institut für Parallele und Verteilte Systeme

Abteilung Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Masterarbeit Nr. 94

Entwicklung eines Frameworks zur Erstellung von AR Anwendungen für Industrie 4.0

Tobias Korb

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	Dr. rer. nat. Frank Dürr
begonnen am:	01.04.2016
beendet am:	04.10.2016
CR-Klassifikation:	H.3.0, H.5.1, H.5.2

Entwicklung eines Frameworks zur Erstellung von AR Anwendungen für Industrie 4.0

Masterarbeit

Autor: **Tobias Korb**
Matrikelnummer: 2679882

Betreuung: Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen

M. Sc. Philipp Sommer

Dr. Ing. Jan Schlechtendahl

Universität Stuttgart

Studiengang: Softwaretechnik

Datum: 01.10.2016

Inhaltsverzeichnis

ABKÜRZUNGEN

ADS - Automation Device Specification

API – Application Programming Interface

AR – Augmented Reality

CAD – Computer Aided Design

CEO – Chief Executive Officer

ERP – Enterprise Resource Planning

GPS – Global Positioning System

LED – Light Emitting Diode

NC – Numeric Control

MES – Manufacturing Execution System

OPC UA – Open Platform Communications Unified Architecture

RFID – Radio Frequency Identification

SaaS – Software as a Service

SDK – Software Developing Kit

SGAM – Smart Grid Architecture Model

SPS – Speicherprogrammierbare Steuerung

URL – Uniform Resource Locator

VR – Virtual Reality

1 EINLEITUNG

In diesem Kapitel zeigt eine Motivation den möglichen Einsatz von Augmented Reality Anwendungen in der Industrie sowie die zusammenhängenden Probleme bei deren Erstellung. Anschließend werden in der Zielsetzung die Probleme bei der Erstellung von Augmented Reality Anwendungen aufgegriffen, um zu definieren, wie diese gelöst werden können. Abschließend werden die vorgegebenen Anforderungen an die Lösungsumsetzung aufgelistet.

1.1 MOTIVATION

Die vierte industrielle Revolution beschreibt die Vernetzung von Menschen, Maschinen und deren Produkten. Einer der ersten Schritte zum Erreichen dieses Ziels besteht darin, die industrielle Produktion mit Hilfe von moderner Informationstechnik zu erweitern. In der digitalen Produktion werden aktuelle Technologien genutzt, um relevante Produktionsdaten digital zu analysieren und auf deren Basis, Verbesserungen in der tatsächlichen Produktion zu erreichen. Um aus Ergebnissen der Analyse von Daten einen Mehrwert generieren zu können, müssen die Resultate dementsprechend in die reale Produktion zurückgeführt werden. Augmented Reality bietet eine Möglichkeit dieses Problem zu lösen. Die Anreicherung der Realität mit Ergebnissen der computergestützten Analyse von Daten kann Prozesse, in denen Menschen involviert sind vereinfachen und somit die Effizienz in der Produktion steigern. Abbildung 1 zeigt eine beispielhafte Anwendung von Augmented Reality in der Produktion. Ein bereits zum Fräsen verwendetes Blech bietet Flächen für weitere Fräsvorgänge. Eine Augmented Reality Applikation analysiert die Fläche und füllt diese mit hinterlegten Fräskonturen aus. Der Bearbeiter sieht somit, wie er das Blech optimal nutzen kann, um unnötigen Ausschuss zu vermeiden.

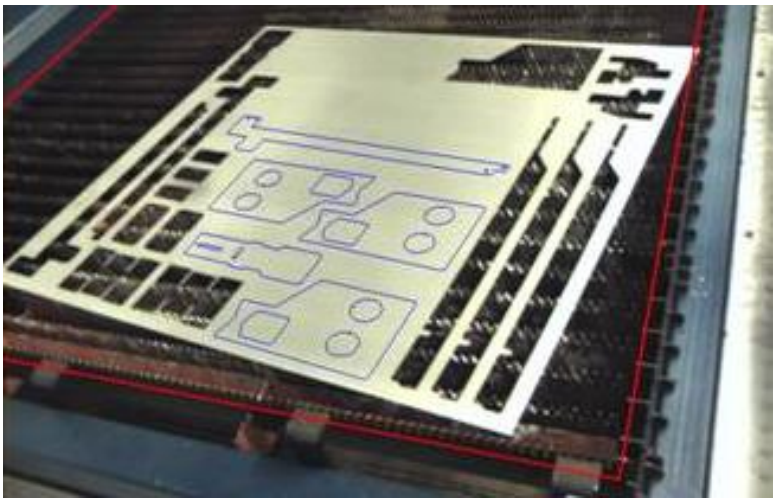

ABBILDUNG 1 - POTENTIELLE NUTZUNG VON AUSSCHUSS MATERIAL BEIM FRÄSEN -

QUELLE: TRUMPF

Die praktische Umsetzung dieser Idee scheitert jedoch häufig an dem Aufwand, produktionsrelevante Daten aus der Maschine mit prozess- und projektbeschreibenden Unterlagen zu verbinden und Ergebnisse in eine Augmented Reality Anwendung zu überführen. Für diese Aufgabe sind sowohl Kenntnisse über Fertigungseinrichtungen und Werkzeugmaschinen, als auch Erfahrung im Bereich der Entwicklung von Augmented Reality Anwendungen nötig. Die seltene Kombination dieser Fähigkeiten und der zeitliche Aufwand zum Erstellen solcher Applikationen erschwert eine praktische Nutzung von Augmented Reality in der Produktion.

1.2 Zielsetzung

In dieser Arbeit wird ein Framework entwickelt, das Entwickler von AR Applikationen für die Industrie dabei unterstützt, Produktionsdaten zu analysieren und diese in Form einer Anwendung rückzuführen. Das Framework bietet dabei Unterstützung in mehreren Teilprozessen der digitalen Produktion. Dieses Framework wird in den weiteren Teilen dieser Arbeit AR-Framework genannt.

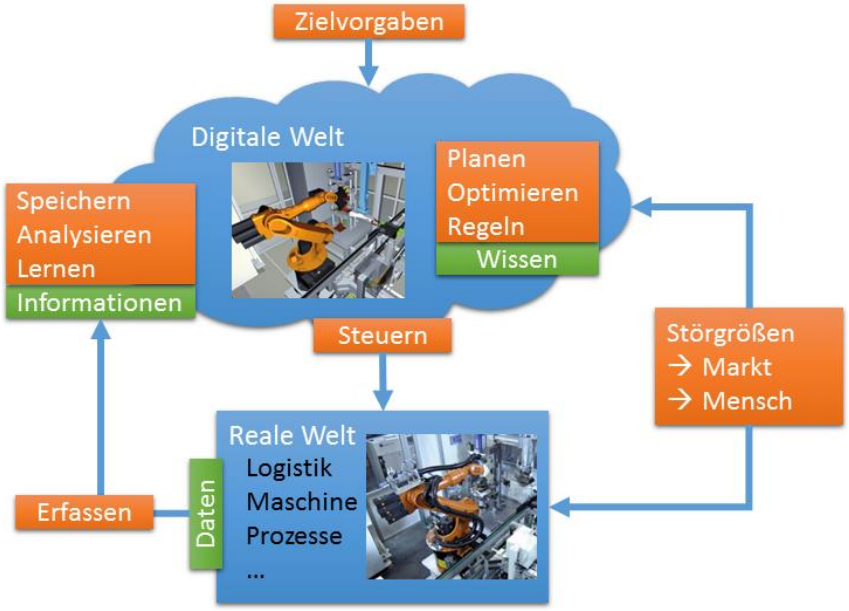

Abbildung 2 - Ablauf der digitalen Produktion – Quelle: Institut für Steuerungstechnik Universität Stuttgart, überarbeitete Version

Abbildung 2 zeigt die grundlegende Idee hinter der digitalen Produktion. Im ersten Schritt müssen reale Daten aus der Produktion erfasst werden. Da diese Daten in unterschiedlichen Formaten vorliegen und durch unterschiedliche Technologien erfasst werden können, soll das AR-Framework eine einheitliche Möglichkeit bieten, verschiedene Arten von Daten zu beziehen und zu verwalten. Live-Daten aus der Maschine, projektbeschreibende Daten aus standardisierten Industrie 4.0 Dokumenten sowie Modellierungsdaten, wie CAD oder UML Dateien sind beispielhafte Daten mit denen das Framework kompatibel sein soll. Das Beziehen sowie das Verwalten dieser Daten soll einfach und standardisiert zu bewerkstelligen sein. Module für einen bestimmten Datentyp innerhalb des Frameworks sollen mit Rücksicht auf Wiederverwendbarkeit entworfen werden. Ein Modul für Maschinendaten soll zum Beispiel sowohl über OPC UA, als auch über ADS erhaltene Daten verwalten können.

Die erfassten Daten müssen im zweiten Schritt analysiert werden, um sie gewinnbringend einsetzen zu können. Diese Analyse soll durch das Framework unterstützt werden, indem es durch einen einheitlichen Zugriff auf die Daten den Umgang erleichtert. Zusätzlich soll das Framework die Möglichkeit bieten, unterschiedliche Daten zu kombinieren, um einen Mehrwert generieren zu können. Module für die Analyse von Daten sollen lose gekoppelt sein, damit ein Modul für unterschiedliche Daten verwendet werden kann. Ein Modul, das Informationen aus einem Dokument an einer bestimmten Stelle im Maschinenkoordinatensystem anzeigt, soll unabhängig vom Datentyp des Dokuments betriebsfähig sein.

Das erlangte Wissen aus der digitalen Welt muss im letzten Schritt wieder in die reale Welt zurückgeführt werden, um einen tatsächlichen Nutzen zu erhalten. Das AR-Framework soll eine

Möglichkeit bieten, das generierte Wissen anhand einer Augmented Reality Lösung zurück in die Produktion zu führen. Augmented Reality Lösungen als neue Mensch-Maschinen-Schnittstellen könnten die Störgröße Mensch aus Abbildung 2 minimieren. Dafür sollen die Informationen in einer Entwicklungsumgebung für Augmented Reality Applikationen bereitgestellt werden. In dieser Entwicklungsumgebung soll ein Entwickler, vom Framework unterstützt die Möglichkeit haben, eine Augmented Reality Anwendung zu entwickeln, die sich an den Mehrwertdaten bedienen kann.

In Abbildung 3 werden die genannten drei Schritte übersichtlich abgebildet und die Art der Unterstützung des Frameworks verdeutlicht.

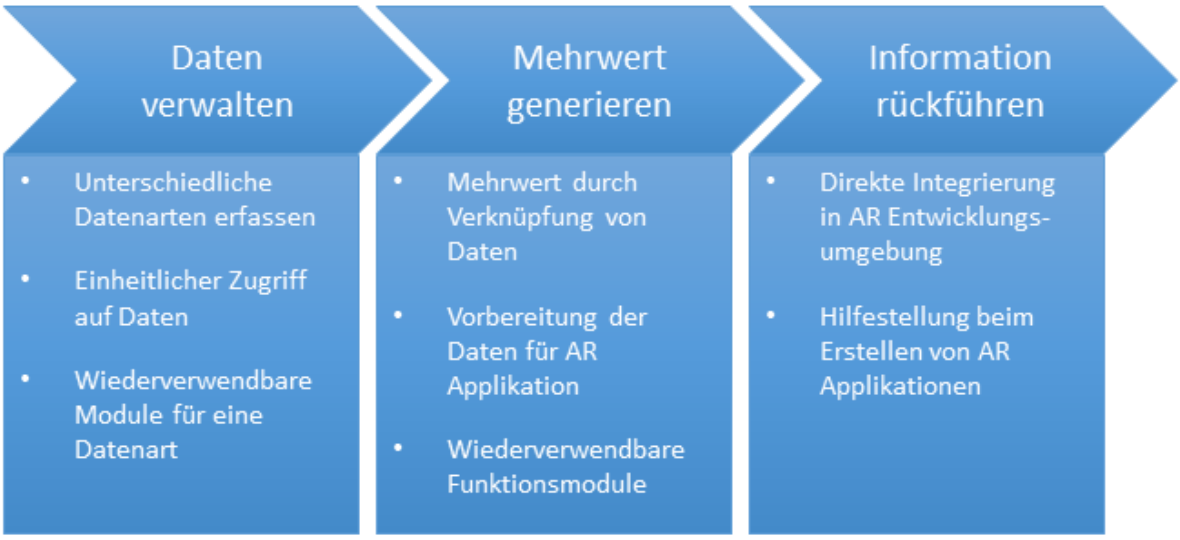

ABBILDUNG 3 - DURCH DAS FRAMEWORK UNTERSTÜTZTE SCHRITTE DER DIGITALEN PRODUKTION

Der folgende Leitsatz fasst den Umfang und den Nutzen des zu erstellenden Frameworks zusammen:

Orientierung:

Ein Entwickler wird vom AR-Framework durch den kompletten Zyklus von der **Erfassung verschiedener Daten**, über die **Datenanalyse und Mehrwertgewinnung** bis hin zur **Rückführung der Daten** in die Produktion begleitet.

Nach der Konzipierung und Implementierung des AR-Frameworks soll evaluiert werden, ob es die im Leitsatz genannten Eigenschaften erfüllt. Dafür wird mit Hilfe des AR-Frameworks eine industrienahe Anwendung in Kooperation mit einem Unternehmen aus der Werkzeugmaschinenbranche entwickelt. Der genaue Aufbau und Zweck der Anwendung wird in Kapitel 6 im Detail erklärt.

1.3 ANFORDERUNGEN

Die Anforderungen an das AR-Framework sind gemäß softwaretechnischer Richtlinien in funktionale und nichtfunktionale Anforderungen unterteilt. Funktionale Anforderungen betreffen direkt den Funktionsumfang des Frameworks. Nichtfunktionale Anforderungen beziehen sich auf Eigenschaften, die die Qualität des Frameworks erhöhen, ohne den Funktionsumfang zu ändern.

Zusätzlich werden alle Anforderungen in die Kategorien „Pflichtanforderung“ und „Zusatzanforderung“ aufgeteilt. Pflichtanforderungen müssen vom Framework erfüllt werden.

Zusatzanforderungen sind optionale Anforderungen, die nach Möglichkeit in der Entwicklung berücksichtigt werden.

1.3.1 Funktionale Anforderungen

Pflichtkriterien

PFA10 Das Framework ermöglicht Mehrwertdiensten Zugriff auf Daten aus verschiedenen Quellen.

PFA11 Die Daten werden durch Hilfsfunktionen aufbereitet und verknüpft um zusätzlichen Mehrwert abzuleiten.

PFA12 Das Framework unterstützt den Entwickler bei der Erstellung von Augmented Reality Anwendungen.

PFA13 Zur Entwicklung neuer Modulbausteine stellt das Framework ein Template zur Verfügung.

PFA14 Zur Entwicklung von AR Anwendungen mit Hilfe des Frameworks und zum Einrichtung der virtuellen Umgebung wird die Game-Engine Unity verwendet.

Wunschkriterien

PFA15 Mögliche bereits umzusetzende Datenmodule sind: Schaltpläne (EPLAN), Steuerungsdaten (OPC UA, Achsdaten, NC-Programme), Planungs- und Steuerungsdaten (ERP, MES), Geometrieinformationen (CAD), Dokumente (Handbücher, Anleitungen)

PFA16 Das Framework stellt vorgefertigte Funktionalität zum Entwickeln von AR Anwendungen zur Verfügung.

1.3.2 Nichtfunktionale Anforderungen

Pflichtkriterien

PNA10 Um die Nutzung des Frameworks zu erleichtern, wird die Funktionalität in Module gekapselt.

PNA11 Das Framework bietet dem Nutzer durch Simplizität und Konsistenz einen einfachen Umgang.

PNA12 Das Framework hält sich an Richtlinien, um ein qualitativ hochwertiges Produkt zu erstellen.

PNA13 Mit dem Framework erstellte AR Anwendungen sind auf verschiedenen Geräten (PC mit Webcam, Smartphone, Tablet, AR-Brille) lauffähig.

PNA14 Das Framework liegt in Form einer Bibliothek zur Nutzung bereit.

PNA15 Die Umsetzung einer prototypischen Anwendung zeigt die Korrektheit des Frameworks.

2 GRUNDLAGEN

In diesem Kapitel werden nötige Grundlagen zum Verständnis der restlichen Arbeit gelegt. Als Leitfaden wird hierbei der Titel der Arbeit verwendet. Der Titel beinhaltet vier verschiedene Gebiete, deren Grundlagen in diesem Kapitel erläutert werden:

Titel der Arbeit:
Entwicklung eines Frameworks zur Erstellung von AR Anwendungen für Industrie 4.0

Die grundlegende Idee von **Industrie 4.0** bildet den ersten Teil dieses Kapitels. Anschließend wird der Begriff Augmented Reality (**AR**) im Kontext Mixed Reality erklärt. Die **Erstellung von Anwendungen** mit Hilfe von Unity wird im dritten Abschnitt dieses Kapitels behandelt. Der letzte Abschnitt beschäftigt sich mit ausgewählten und in dieser Arbeit umgesetzten Mustern zur **Entwicklung von Frameworks** mit hoher Qualität.

2.1 INDUSTRIE 4.0

Industrie 4.0 ist ein aus Deutschland stammender Begriff, der für die Einleitung der vierten industriellen Revolution steht. Nachdem der Einsatz von Informationstechnik in der Industrie zur Automatisierung von Prozessen seit Jahren genutzt wird, sollen im Rahmen von Industrie 4.0 aktuelle Internet-Technologien genutzt werden, um die Vernetzung innerhalb der produzierenden Industrie weiter zu steigern. Mit steigender Vernetzung und somit steigender Kommunikation zwischen den einzelnen Komponenten können produktionsrelevante Informationen intelligent geteilt und digital verarbeitet werden. Durch diese digitale Verarbeitung von Daten aus der Industrie entsteht Wissen, das die zukünftige Produktion optimieren kann. Neben höherer Qualität oder Optimierung von Prozessen könnten laut Bauernhansel, ten Hompel und Vogel-Heuser [1] komplett neue Geschäftsmodelle entwickelt werden.

Der Fachausschuss 7.21 „Industrie 4.0“ der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik unter der Leitung von Prof. Dr.-Ing. Ulrich Epple von der RWTH Aachen versucht exakte Konventionen und Standards für Industrie 4.0 zu entwickeln, um dies zu erreichen. Zum Zeitpunkt dieser Arbeit resultierte aus den Bemühungen dieses Gremiums das Referenzarchitekturmodell RAMI 4.0 [2]. In dieser Referenzarchitektur werden grundlegende Namenskonventionen sowie ein in Abbildung 4 dargestelltes, kubisches Schichtenmodell zur Beschreibung von Industrie 4.0 Komponenten definiert.

Das Schichtenmodell ist an das Smart Grid Architecture Model (SGAM) angelehnt. Entlang der vertikalen Achse des 3D Modells werden die verschiedenen Abstraktionen der Sichtweise auf die Industrie auf sechs Ebenen verteilt. Die Business Ebene bezieht sich auf Unternehmensrelevante Ideen wie zum Beispiel die IT Strategie eines Unternehmens. Das andere Extrem, die Asset Ebene beschreibt physikalisch vorhandene Elemente wie einzelne Werkstücke oder Dokumente. Die Value Stream Achse des Modells beschreibt den kompletten Lebenszyklus eines Produkts von der Konzeption bis zur Wartung. Dieser zeitliche Ablauf lässt sich in die Typ- und Instanzphasen unterteilen. Ein Produkttyp beschreibt die Idee und den Aufbau eines bestimmten Produkts. Angefangen von der Idee bis hin zu ersten Prototypen. In der zweiten Phase wird das Produkt produziert und somit eine Instanz des Typs gefertigt. Die Hierarchielevels komplettieren das Architekturmodell. Diese Achse ordnet Informationen zusätzlich zu den Ebenen und dem Punkt im Lebenszyklus, dem funktionalen Betrachtungslevel zu. Von dem hergestellten Produkt an sich bis hin zu außerbetrieblichen Komponenten wie Zulieferern und Kunden.

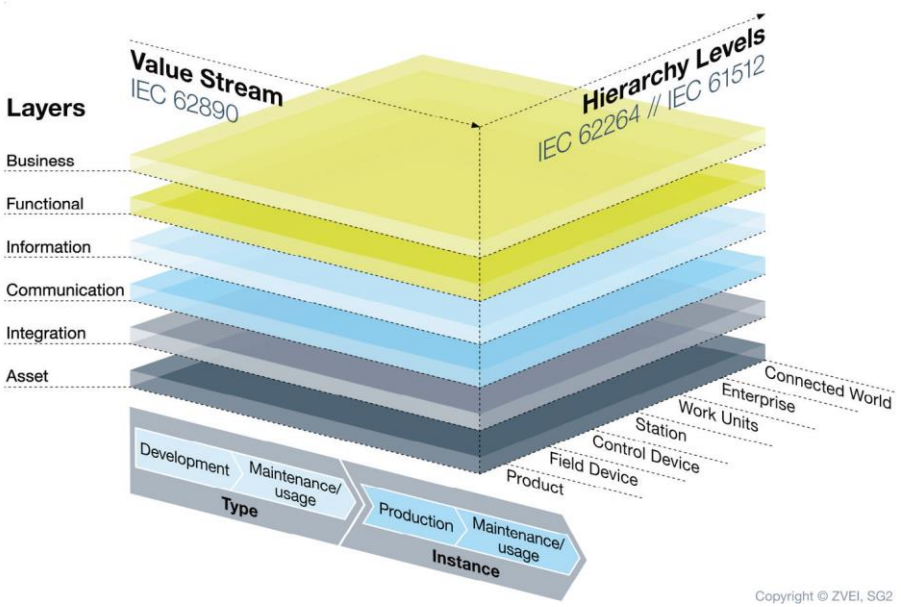

ABBILDUNG 4 - SCHICHTENMODELL DER REFERENZARCHITEKTUR FÜR INDUSTRIE 4.0 [2]

Werden die Standards, die in dieser Referenzarchitektur deutlich tiefreichender als in diesem Kapitel beschrieben sind, von der Industrie umgesetzt, führt dies zu standardisierten Vorgehensmodellen, Beschreibungen und Produkten. Dies ermöglicht die gewünschte Vernetzung der Produktion und damit die Gewinnung von Produktionsdaten zur Analyse.

2.2 AUGMENTED REALITY

Eine Einordung und Erklärung von Augmented Reality wurde 1994 von Paul Milgram [3] vorgenommen. In seinem Realitäts-Virtualitäts-Kontinuum präsentiert Milgram [3] eine Skala zur Beschreibung von realitätserweiternden technologischen Konzepten. Die beiden äußeren Blöcke in Abbildung 5 beschreiben die ausschließlich reale und die ausschließlich virtuelle Welt. Zwischen den beiden Extremen befindet sich die gemischte Realität (Mixed Reality). In diesem Bereich verschwimmt die echte Welt mit der virtuellen Welt. Die Mixed Reality wird in die Erweiterte Realität (Augmented Reality) und die Erweiterte Virtualität (Augmented Virtuality) unterteilt.

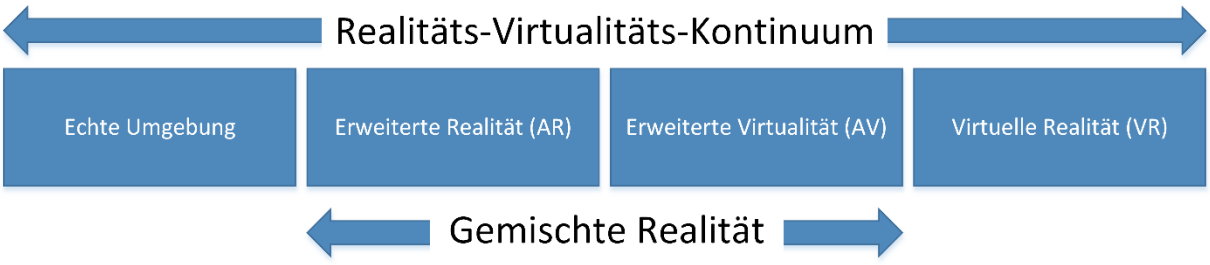

ABBILDUNG 5 - REALITÄTS-VIRTUALITÄTS-KONTINUUM NACH MILGRAM [3]

Die Idee hinter dem Konzept der Augmented Virtuality ist die Erweiterung der virtuellen Welt mit realen Ausschnitten. Eine über Video aufgenommene Person, die in einer rein virtuellen Welt dargestellt wird, fällt in diese Kategorie der gemischten Realität. Augmented Reality verfolgt den umgekehrten Ansatz. Die reale Welt wird mit virtuellen Elementen angereichert. Um dies zu erreichen, muss die reale Welt analysiert und virtuelle Gegenstände realitätsnah in dieser Welt untergebracht werden. Hierbei ergeben sich verschiedene Probleme, die gelöst werden müssen. Es ist nötig, im

dreidimensionalen Raum zu erkennen, wo ein Element platziert werden soll. Je nach Betrachtungswinkel muss dieses Element entsprechend verzerrt dargestellt werden, um möglichst real zu erscheinen. In Abbildung 6 werden drei Quader von zwei unterschiedlichen Blickwinkeln betrachtet. Da die Quader an den Blickwinkel der betrachtenden Person angepasst und entsprechend verzerrt sind, erstellt das menschliche Gehirn in beiden Fällen ein dreidimensionales Bild. Es ist entsprechend nötig zu wissen, wo ein Element angezeigt wird und von wo aus es betrachtet wird, um diesen Effekt zu produzieren.

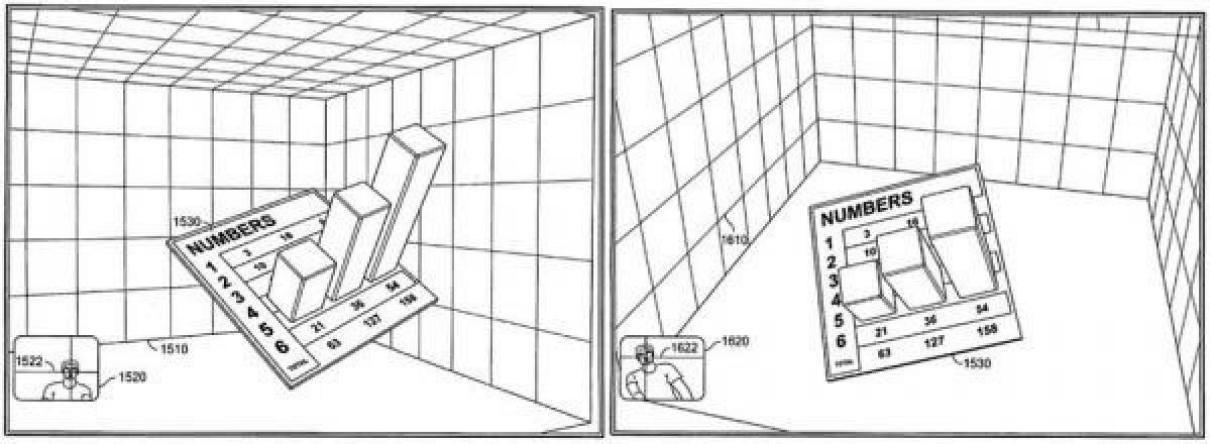

Abbildung 6 - Verzerrte Darstellung von Quadern relativ zur Kamera - Quelle: Macrumors[1]

In einem abgeschlossenen Raum werden hierfür nach You, Neumann und Azuma [4] zwei verschiedene Techniken verwendet. Für beide Techniken sind die gleichen Komponenten nötig. Zum einen ist eine Markierung für die Position der virtuellen Elemente nötig und zum anderen ein Sensor für die Erkennung dieser Markierungen. Bevor beide Techniken erklärt werden können, ist eine Erklärung der Marker und der Sensoren nötig.

Die Art der Markierung zum Platzieren des Objekts ist abhängig vom genutzten Sensor und umgekehrt. Wenn der Sensor beispielweise eine Kamera ist, kann die Markierung der Position keine Magnetfeldquelle sein. Ein bewährtes Mittel für Kamerasensoren bieten sogenannte passive Marker. Passive Marker sind optisch erkennbare Muster, die innerhalb des Raums platziert werden. Sie enthalten häufig Symbole in schwarz und weiß, da der hohe Kontrast zwischen diesen Farben und ihrer Umwelt maschinell leicht zu erkennen ist. Ein Vorteil dieser Methode ist die Möglichkeit der sehr genauen Platzierung eines virtuellen Gegenstands im echten Raum, da die Marker-Koordinaten zur Positionierung genutzt werden können. In Abbildung 7 sind zwei beispielhafte passive Marker zu sehen. Bei beiden Markern fallen die prägnanten geometrischen Formen auf. Diese helfen den Betrachtungswinkel der Kamera zu berechnen. Passive Marker müssen nicht immer aus größtenteils geometrischen Formen bestehen. Ein unbearbeiteter fünf Euro Schein kann ebenfalls als Marker verwendet werden. Dieser bietet durch die oben genannten fehlenden Eigenschaften eine schlechtere Qualität als die Marker aus Abbildung 7, ist andererseits jedoch für bestimmte Anwendungsfälle nützlicher. Soll zum Beispiel ein virtuelles Objekt auf dem Sicherheitsglas einer Werkzeugmaschine angezeigt werden, kann als Marker das Logo des Maschinenherstellers anstatt einem auffälligen

[1] http://www.macrumors.com/roundup/apple-vr-project/

Schwarz-Weiß-Marker verwenden werden. Dieses Vorgehen führt zu einer höheren Immersion des Maschinenbedieners.

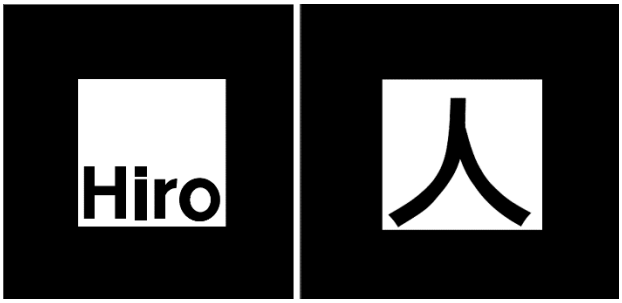

ABBILDUNG 7 - PASSIVE MARKER DES ARTOOLKITS - QUELLE: BELAJAR-AR[2]

Im Vergleich zu passiven Markern, geben aktive Marker ein Signal ab. Der Signaltyp ist nicht genau spezifiziert. Ein Beispiel für einen aktiven Marker ist eine leuchtende LED, ein anderes Beispiel ein Gegenstand der Magnetwellen aussendet. Für einen aktiven Marker können neben einer Kamera auch andere Sensoren verwendet werden. Beispielhafte Sensoren sind Empfänger für Magnetwellen oder für aktuelle Technologien wie RFID. GPS im Inneren eines Gebäudes zu nutzen, liefert selten gute Ergebnisse, weshalb diese Technologie laut Bostanci et al. [5] meist nur in Außenbereichen eingesetzt wird. Aktive Marker benötigen in den meisten Fällen leistungsstärkere Hardware als passive Marker und sind somit für mobile Endgeräte laut Cheok [6] und Golding [7] nur selten geeignet.

Die zwei Techniken zum Erkennen der Marker unterscheiden sich in der Art der Platzierung von Marker und Sensor. Bei der Outside-In Methode befindet sich der Sensor an einer statischen Stelle im Raum und dementsprechend außerhalb (Outside) des Benutzers. Die Marker hingegen werden direkt am Nutzer angebracht (In). Die Inside-Out Methode funktioniert auf umgekehrte Weise. Der Sensor wird am Nutzer angebracht (Inside) und bewegt sich somit frei im Raum. Die Marker sind im Raum verteilt und somit außerhalb des Nutzers positioniert (Out). Durch beide Techniken erhält man die Position des anzuzeigenden virtuellen Elements sowie den Betrachtungswinkel und kann somit das Element platzieren und die echte Welt virtuell erweitern.

2.3 UNITY

Unity ist eine Laufzeitumgebung für aktuelle Plattformen wie PC, Smartphones und Spielkonsolen. Die Entwicklung von Anwendungen für Unity erfolgt durch die Entwicklungsumgebung, die ebenfalls den Namen Unity trägt. Beide Projekte stehen zur kostenlosen kommerziellen Nutzung bereit, solange das Unternehmen einen Umsatz von unter 100.000 Euro im Jahr vorweist [8]. Durch die Zusammenarbeit von Unity Technologies mit Valve und HTC, ist Unity laut CEO John Riccitiello [9] die derzeit erfolgreichste Plattform zur Entwicklung von Virtual Reality Applikationen. Aktuelle Augmented Reality Toolkits wie ARToolKit und Vuforia setzen ebenfalls auf Unity als Entwicklungsplattform für Augmented Reality Software. Microsoft kooperiert mit Unity, um die Entwicklungsumgebung für die Augmented Reality Brille HoloLens anzupassen [10]. In Abbildung 8 ist die grundlegende Struktur von Unity abgebildet. Auf die einzelnen Komponenten wird im weiteren Verlauf näher eingegangen.

Die auf OpenGL und Direct3D basierende Grafik Engine in Unity unterstützt aktuelle Techniken wie dynamische Schatten oder Postprocessing Effekte, um virtuelle Gegenstände realitätsnah aussehen zu lassen. Physikalische Berechnungen von Elementen in Unity übernimmt PhysX für 3D-Elemente und

[2] http://belajar-ar.blogspot.de/2010/05/tutorial-artoolkit-part-1-installing.html

Box2D für 2D-Elemente. Lichtquellen und das Verhalten von Licht beim Treffen auf virtuelle Elemente werden seit 2015 mit Enlighten berechnet. Zusätzlich zu den grafikorientierten Technologien, beinhaltet die Unity Engine die Bibliothek FMOD, die für Töne und Klänge in Unity Anwendungen sorgt. Die weitere Basisfunktionalität, die das Gerüst zwischen den einzelnen Technologien bildet und die einzelnen Komponenten der Entwicklungsumgebung verbindet, ist nicht öffentlich zugänglich.

Damit der Nutzer von den Technologien hinter Unity profitieren kann ohne sich mit den Einzelheiten der Unity Engine auseinander setzen zu müssen, stellt Unity eine Entwicklungsumgebung zur Verfügung. Der in der Entwicklungsumgebung eingebaute 3D-Editor bietet dem Entwickler eine Möglichkeit, erstellte Modelle in einem 3D-Raum anzuordnen oder zu editieren. Komplexere Modelle können aus 3D-Modellierungsprogrammen wie 3ds Max oder von CAD-Modellen importiert werden. Die Entwicklungsumgebung beinhaltet jedoch keine Möglichkeit, den virtuellen Elementen direkt im 3D-Editor eine Logik zu hinterlegen. Die Logik-Komponente in Unityprogrammen wird strikt von der 3D-Modellierung getrennt. Durch die starke Modularisierung und der Möglichkeit schwach gekoppelte Programme zu entwickeln, können in Unity erstelle Szenen oder Modelle häufig wiederverwendet werden.

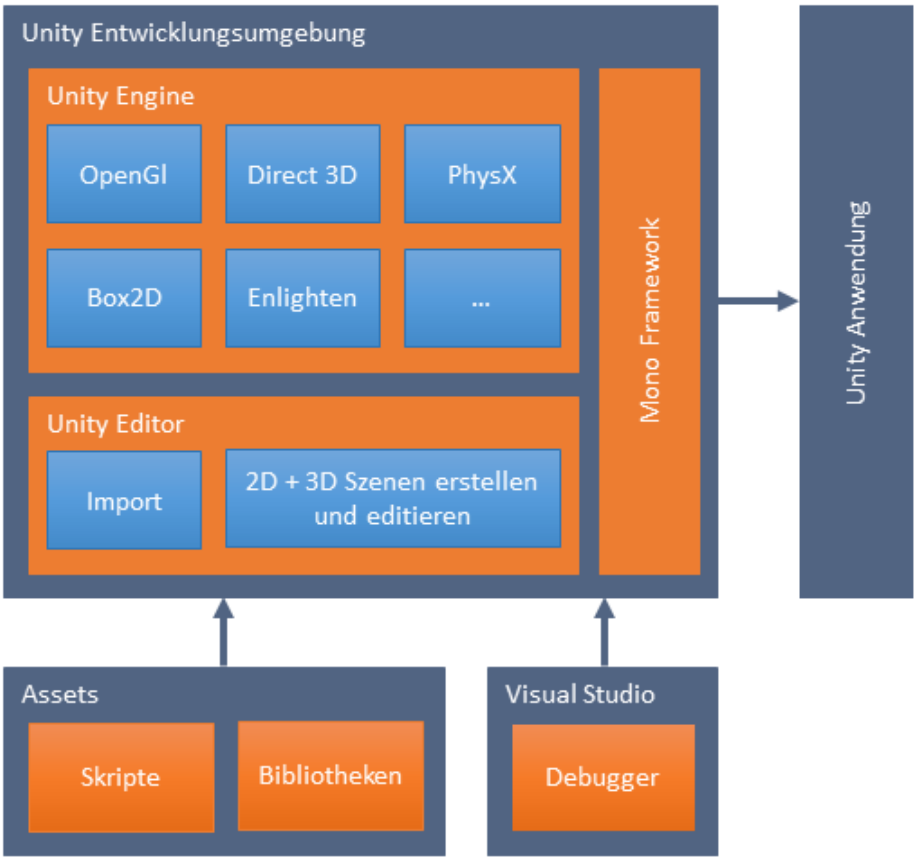

ABBILDUNG 8 - ABSTRAKTER AUFBAU DER KOMPONENTEN DER UNITY ENTWICKLUNGSUMGEBUNG

Um einem erstellten, virtuellen Gegenstand eine Logik zu hinterlegen, werden Skripte verwendet, die mit hohen Programmiersprachen wie C# geschrieben sind. Diese Skripte können in Unity einzelnen Objekten in einer Szene zugeordnet werden und kontrollieren somit das Verhalten dieses Objekts. An einem Objekt können mehrere Skripte angebracht werden, die bei der Ausführung parallel abgearbeitet werden. Ein Entwickler kann Skripte in einer beliebigen Entwicklungsumgebung erstellen, bevor sie in Unity eingebunden werden. Durch eine Kooperation mit Microsoft Visual Studio bietet

Unity die Möglichkeit zur Laufzeit, Debuginformationen an Visual Studio zu senden. Da C# Skripte nicht nativ auf jeder Plattform ausführbar sind, nutzt Unity das Mono Framework. Dieses Framework ist eine plattformunabhängige Umsetzung des .NET Frameworks von Microsoft. Durch die Nutzung von C# mit Mono, kann ein Unity Skript auf Bibliotheken des .NET Frameworks zugreifen und diese nutzen. Diese Nutzung von externen Bibliotheken erlaubt eine übersichtliche Strukturierung innerhalb der Skripte. Ein Skript in Unity hat zwei standardmäßig zu implementierende Methoden. Eine Methode wird beim Starten der Anwendung einmal ausgeführt. Die andere Methode wird bei jeder Aktualisierung des Bildes ausgeführt. Dementsprechend besteht eine Relation zwischen den Funktionsaufrufen und der Bildfrequenz der Anwendung. Neben diesen beiden Kernmethoden bietet Unity noch weitere Methoden an, die in Skripten genutzt werden können, um auf Geschehnisse im Unitprogramm zu reagieren. Auf diese Weise kann beispielsweise eine Methode speziell vor oder nach dem Rendern des Bildes aufgerufen werden, ohne diesen Zeitpunkt selbst ermitteln zu müssen.

Durch die Unterstützung vieler großer Firmen, der aktuellen Position als Marktführer im Bereich VR-Entwicklung und der Tatsache, dass Unity unter einer kostenlosen Lizenz für kleine kommerzielle Anwendungen veröffentlich wurde, ist es sehr wahrscheinlich, dass Unity auch in naher Zukunft eine relevante Stellung im Augmented Reality Bereich einnehmen wird. Aus diesem Grund setzt das, in dieser Arbeit entwickelte AR-Framework auf Unity als Entwicklungsplattform und Laufzeitumgebung für Augmented Reality Anwendungen.

2.4 Framework Qualität

Als Framework bezeichnet man in der Informatik ein Gerüst, das von Entwicklern genutzt wird, um ein Produkt zu entwickeln. Als Fundament für mehrere Anwendungen sollte ein Framework entsprechend eine hohe Qualität bieten. Um dies zu gewährleisten, werden in dieser Arbeit Framework Patterns genutzt. Ein Pattern ist eine Vorlage zur Lösung eines bestimmten Problemtyps. Diese Vorlage bietet keine sofort einsetzbare Implementierung für ein gewisses Problem, sondern ein Konzept, das für eigene Implementierungen genutzt werden sollte. Cwalina und Abrams haben in ihrem Buch *„Framework Design Guidelines“* [11] eine hochwertige Sammlung von Framework Patterns veröffentlicht. Zusätzlich bietet das Buch Qualitätsmerkmale zum Bewerten eines Frameworks. Microsoft hat in der Dokumentation für das .NET Framework große Teile des Buches als offizielle Richtlinien zur Erstellung von Frameworks veröffentlicht [12]. Nachfolgend werden wichtige Qualitätsmerkmale eines Frameworks und Pattern zur Gewährleistung dieser Merkmale vorgestellt.

2.4.1 Qualitätsmerkmale eines Frameworks

Damit ein Entwickler sich entscheidet, ein Framework zu nutzen, muss es ihm einen theoretischen Mehrwert für seine Arbeit bringen. Existiert dieser theoretische Mehrwert, muss er auch abgerufen werden können. In vielen Fällen erkennt ein Entwickler erst mit der Nutzung eines Frameworks, dass es einige Probleme behebt, jedoch auch neue verursacht. In den meisten Fällen geschieht dies laut Cwalina und Abrams [11] nicht durch funktionale Probleme des Frameworks, sondern durch konzeptionelle Fehlentscheidungen. Aus diesem Grund sollte ein Framework einige grundlegende Eigenschaften besitzen, die es dem Nutzer erlauben, den theoretischen Mehrwert in vollem Umfang aus einem Framework zu beziehen. Die folgenden Eigenschaften sind eine ausgewählte Teilmenge der von Cwalina und Abrams [11] definierten Qualitätsattribute.

Software Frameworks sind einfach!

Ein Framework hat die Aufgabe, einen Entwickler bei seiner Tätigkeit zu unterstützen. Sollte sich der Nutzer neben seiner Hauptaufgabe noch intensiv mit dem Framework beschäftigen müssen, verliert das Framework bereits seinen Nutzen. Ein gutes Framework muss deshalb einfach sein. Instinktiv denkt man bei einem einfachen Framework an die einfache Benutzung des Frameworks. Dies ist aber nur einer von vielen Teilen, die ein Framework als Ganzes einfach macht. Bevor ein Entwickler ein Framework benutzen kann, muss er das Framework verstehen. Er muss verstehen, welchen Mehrwert ihm das Framework bietet, aber auch welche Funktionalität das Framework nicht bietet. Diese Aufgabe wird von einem Handbuch gelöst, das beide Punkte von Anfang an offen auflisten sollte. Hat sich ein Entwickler für ein Framework entschieden, wird das Framework vor der Nutzung eingerichtet. Um diesen Schritt einfach zu gestalten, sollte das Framework einen begleitenden Installationsprozess bieten. Selbst der Prozess, verschiedene Dateien in unterschiedliche Ordner zu legen, kann durch einen Installationsprozess vermeiden, dass Fehler auftreten. Ist das Framework installiert, muss ein Framework häufig für spezifische Aufgaben angepasst oder erweitert werden. Diesen Prozess so einfach wie möglich zu gestalten, bietet eine große Herausforderung. Aus diesem Grund widmen Cwalina und Abrams [11] dieser Aufgabe ein eigenes Kapitel, das im folgenden Abschnitt zusammengefasst wird.

Software Frameworks sind für Erweiterbarkeit designt!

Die Grundidee für eine hohe Erweiterbarkeit liegt in einem modularen Design aller wichtigen Komponenten des Frameworks. Bietet ein Framework zum Beispiel verschiedene Kommunikationsmechanismen an, ist es sehr wahrscheinlich, dass ein neuer Mechanismus hinzugefügt wird. Es bietet sich entsprechend an, die Mechanismen in ein modulares System aufzuteilen und eine globale Schnittstelle anzubieten, an denen die verschiedenen Kommunikationsmechanismen eingebunden werden können. Diese Überlegung sollte für alle Funktionen des Frameworks getroffen werden. Sobald ein Framework eine Aufgabe auf mehrere Arten lösen kann, bietet ein modulares Design die beste Möglichkeit für eine Erweiterung.

Um dieses Softwaredesign zu unterstützen und die Erweiterung noch simpler zu gestalten, sind Vorlagen für neue Module nötig. Eine mögliche Variante sind zu implementierende Schnittstellen, die einerseits die Integrierung der Module in das Framework erleichtern und andererseits dem Entwickler bei den grundlegenden Funktionalitäten eines Moduls weiterhelfen können. Die Arbeit von Cwalina und Abrams [11] bezieht sich explizit auf das .NET Framework. Mit Visual Studio als führende Entwicklungsumgebung für Anwendungen basierend auf dem .NET Framework, besteht die Möglichkeit Klassenvorlagen anzulegen. Diese Vorlagen bieten eine noch einfachere Möglichkeit, ein Framework zu erweitern. Der Entwickler bekommt eine Vorlage, die ihm durch erklärende Kommentare, implementierte Schnittstellen und konventionsgerecht benannte Objekte einen einfachen Start bietet.

Software Frameworks sind konsistent!

Konsistenz innerhalb eines Frameworks ist ein weiterer Faktor zur einfacheren Benutzung. Aus diesem Grund sollte ein Framework auf jedem Abstraktionslevel konsistent sein. Wie bereits beschrieben, beginnt die Arbeit mit einem Framework beim Lesen des Handbuchs. Aus diesem Grund ist es wichtig, das Handbuch auch nach Änderungen an dem Framework konsistent zu halten. Dieses Vorgehen sollte laut Cwalina und Abrams [11] Standard sein. Nach ihrer Erfahrung gibt es jedoch nach einigen

Versionsänderungen der Software nur selten konsistente Handbücher. Auch innerhalb der Software gibt es verschiedene Wege Konsistenz zu erhalten. Gängige Namenskonventionen für Klassen, Schnittstellen, *Enumerations*[3], Variablen und Membervariablen sind für ein Framework Pflicht, um einem Entwickler eine instinktive Nutzung zu ermöglichen. Designentscheidungen wie die Wahl zwischen abstrakten Klassen und Schnittstellen sollten im kompletten Framework konsistent getroffen werden.

Software Frameworks sind dokumentiert!

Die Dokumentation eines Frameworks wurde in diesem Kapitel mehrmals angesprochen. Qualitativ hohe Software verfügt über eine sehr gute Dokumentation außerhalb, sowie innerhalb des Quellcodes. Dies fördert die Überarbeitung oder Erweiterung der Software. Für ein Framework ist die Dokumentation aus diesem Grund noch wichtiger als für jede andere Software. Ohne Erweiterungsmöglichkeiten verliert ein Framework einen Großteil seines Nutzens. Mit einer guten Dokumentation hingegen wird das Fundament für eine gute Erweiterbarkeit gelegt. Zu beachten ist laut Cwalina und Abrams [11], dass die Dokumentation konsistent erstellt wird. Dies wird durch Vorlagen oder unterstützende Technologien möglich gemacht.

Die vier genannten Merkmale sind nur ein Ausschnitt aus den Merkmalen die Cwalina und Abrams [11] präsentieren. Diese vier Abschnitte überschneiden sich teilweise deutlich, was zeigt, dass für eine hohe Qualität eines Frameworks mehrere Merkmale umzusetzen sind und es nicht genügt, sich auf ein Merkmal zu fokussieren. Eine gute Erweiterbarkeit zum Beispiel, ist nur schwer ohne ein konsistentes und dokumentiertes Fundament möglich.

2.4.2 Framework Guidelines

Um die genannten Qualitätsmerkmale zu erfüllen, bieten Cwalina und Abrams [11] Hilfestellungen für die Entwicklung eines Frameworks an. Eine Lösung bietet jeweils eine oder mehrere Vorgehen für ein abstraktes Problem. Für das Framework das in dieser Arbeit entsteht, werden mehrere dieser Hilfestellungen genutzt. Grundlegend sind Hinweise in vier verschiedene Kategorien unterteilt, die in Tabelle 1 abgebildet sind.

Tabelle 1 - Kategorien für Hinweise in „Framework Design Guidelines“ von Cwalina und Abrams [11]

Do	Die Methode sollte immer umgesetzt werden
Consider	Die Methode könnte situationsbedingt die Qualität erhöhen
Avoid	Bis auf wenige Ausnahmen sollte diese Methode vermieden werden
Do not	Diese Methode sollte niemals verwendet werden

Das Buch umfasst über 30 verschiedene Themengebiete, die alle genutzt werden, um das Framework dieser Arbeit zu erstellen. Zu zwei verschiedenen Themengebieten werden nachfolgend beispielhafte Ausschnitte aus der Arbeit von Cwalina und Abrams [11] präsentiert.

Namensgebung

Für eine durchgängig konsistente und intuitive Namensgebung von Klassen, Methoden und Parametern fassen Cwalina und Abrams [11] einige bewährte Praktiken zusammen. Ein Beispiel ist die

[3] Ein in der Informatik genutzter Datentyp für Aufzählungen

korrekte Anwendung von *Pascal*- und *Camelcasing*. Beide Praktiken beschreiben eine Methode zur Groß- und Kleinschreibung von einzelnen oder zusammengesetzten Wörtern beim Programmieren. Beim *Pascalcasing* beginnen alle Wörter mit einem Großbuchstaben gefolgt von Kleinbuchstaben. Beim *Camelcasing* beginnt das erste Wort mit einem Kleinbuchstaben und jedes weitere Wort mit einem Großbuchstaben. Grundsätzlich sollte das *Camelcasing* nur bei Parametern von Methoden genutzt werden. Für alle anderen Klassen, Methoden oder Variablen sollte das *Pascalcasing* verwendet werden. Abbildung 9 zeigt einen beispielhaften Ausschnitt von Hinweisen zum Benennen von Bezeichner.

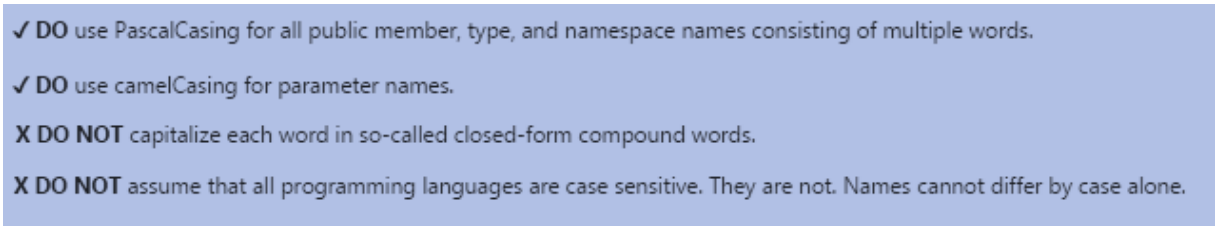

Abbildung 9 - Konventionen zur Namensgebung nach Cwalina und Abrams - Quelle: Microsoft [12]

Die Konventionen für die Groß- und Kleinschreibung von Bezeichnern mögen nicht sonderlich kompliziert klingen. Hält man jedoch alle der über 50 Vorschläge zu diesem Gebiet ein, besitzt der Quellcode eines Frameworks einen konsistenten Aufbau, der einfach und intuitiv zu verstehen ist.

Überladung von Methoden

Neben den oberflächlichen Ratschlägen zur Namensgebung, geht das Buch auch auf tiefergehende Designentscheidungen ein. Das Kapitel zum Überlagern von Methoden zeigt zum Beispiel diverse Techniken, die benutzerfreundliche Methodenaufrufe erlaubt. In Abbildung 10 wird ein Ausschnitt der Ratschläge aus diesem Kapitel vorgestellt. In einem im Juni 2016 veröffentlichten Artikel von Myers und Stylos [13] wird getestet, wie man die Benutzbarkeit von APIs steigern kann. Ihre Erkenntnisse decken sich größtenteils mit den von Cwalina und Abrams [11] vorgeschlagenen Methoden.

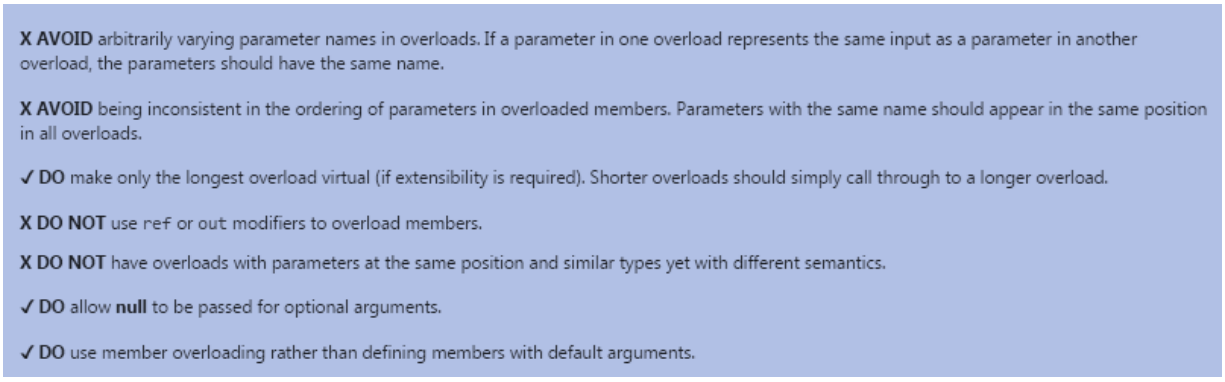

Abbildung 10 - Konventionen zum Überladen von Methoden nach Cwalina und Abrams - Quelle: Microsoft [12]

Die Richtlinien von beiden vorgestellten Themengebieten sorgen für eine höhere Qualität des Frameworks gemessen an den Merkmalen aus Kapitel 2.4.1. Durch die zusätzliche Nutzung der restlichen Vorschläge besitzt das AR-Framework eine hohe Qualität.

3 STAND DER TECHNIK

Für diese Arbeit ist es nötig, den Stand der Technik in zwei unterschiedlichen Bereichen zu betrachten. Der erste Teil umfasst Kapitel 3.1 und 3.2 und beschäftigt sich mit Augmented Reality Themen. In Kapitel 3.1 werden Augmented Reality Technologien untersucht, die aktuell in der Produktion zum Einsatz kommen. In Kapitel 3.2 werden verschiedene Software Development Kits (SDK) betrachtet. AR-SDKs unterstützen bei der Erstellung von Augmented Reality Anwendungen und sind somit auch für das AR-Framework relevant. Um zu entscheiden, ob ein eigenes SDK erstellt werden muss oder ob der Stand der Technik in diesem Bereich den Anforderungen an das AR-Framework genügt, werden die betrachteten SDKs miteinander verglichen. Eine Bewertung der betrachteten Technologien basiert auf ausgewählten Metriken.

Neben der Darstellung von Informationen in Augmented Reality Anwendungen, ist eine weitere Aufgabe des Frameworks, verschiedene Daten zu beziehen und zu verwalten. Der zweite Teil des Stands der Technik befasst sich sin Kapitel 3.3 deshalb mit Datenhaltungstechnologien für relevante Daten aus der Industrie und der Produktion. Hierbei wird Wert auf Methoden gelegt, die einen programmatischen Zugriff auf die Inhalte der Daten erlauben. Abschließend werden die Erkenntnisse des Stands der Technik genutzt, um zu definieren wie diese Arbeit den Stand der Technik erweitern kann.

3.1 AUGMENTED REALITY TECHNOLOGIEN IN DER PRODUKTION

Der Gartner Hype Cycle für aufkommende Technologien für das Jahr 2016 [14] zeigt, dass Augmented Reality die Phase der überhöhten Erwartungen durchlaufen hat und in den kommenden Jahren für eine produktive Nutzung bereit sein wird. Schon jetzt befassen sich verschiedene Projekte mit dem Einsatz von Augmented Reality in der Produktion.

Die Automobilindustrie hat bereits während der Hype-Phase versucht, Augmented Reality in der Produktion zu verwenden. BMW [15] hat getestet ob es möglich ist, mit Hilfe einer Augmented Reality Anwendung die Effizienz bei Schweißvorgängen zu verbessern. VW [16] hat in Kooperation mit Metaio eine Anwendung zur Optimierung von Planungsprozessen entwickelt. Auch Daimler [17] hat im Zuge der ARVIKA Initiative [18] Augmented Reality Anwendungen in der Produktion von Automobilteilen genutzt. General Motors [19] hat 2012 eine Anwendung entwickelt, die darauf basiert, Projektoren anstatt AR-Brillen zu verwenden, um die reale Welt mit Informationen zu füllen. Das Ergebnis dieses Projekts ist eine visuelle Darstellung von Schweißpositionen auf Werkstücken. In der Luftfahrtindustrie hat Airbus [20] den Einsatz von Augmented Reality zum Einblenden von Zusatzinformationen bei der Wartung von Flugzeugteilen getestet. Ein Tablet dient in diesem Versuch als Medium zum Erweitern der Realität.

Hilfestellung bei der Wartung von Produktionsteilen ist ein Themengebiet, in dem Augmented Reality häufig getestet wird. Garza et al. [21] haben CAD Modelle erstellt, die in einer Augmented Reality Anwendung einem Wartungsingenieur zeigen, aus welchen Komponenten das zu wartende Teil besteht. Diese Anwendung wird zur Fort- und Ausbildung von Wartungsingenieuren genutzt. Fiorentino et al. [22] haben 2013 eine Anwendung entwickelt, die bei der Montage und Demontage von Motorradmotoren unterstützt. Die Anwendung zeigt dem Ingenieur die notwendigen Schritte visuell an. Eine Auswertung der Testdaten ergab eine statistisch relevante Produktionssteigerung bei gleichzeitiger Fehlerminderung. Webel et al. [23] haben im Auftrag des Fraunhofer IGD ebenfalls eine Augmented Reality Anwendung zur Ausbildung entwickelt. Zusätzlich zur Anwendung bekommt der Auszubildende ein Armband an die dominante Hand. Die Anwendung kann dem Benutzer über eine

Vibration des Armbands signalisieren, wenn er einen Fehler macht. Auf der CIRP Konferenz 2015 wurde von Syberfeldt et al. [24] eine dynamische Wartungsanwendung präsentiert. Diese Anwendung erweitert zu wartende Teile auf unterschiedliche Weise. Je nach Fähigkeit des Benutzers werden Informationen in unterschiedlichen Detaillevel angezeigt. Azpiazu et al. [25] haben eine Anwendung zur Unterstützung bei der Wartung und Reparatur von Werkzeugmaschinen entwickelt. Anstatt statische Instruktionen an der Maschine einzublenden, wird eine Videoaufnahme direkt an einen geschulten Monteur gesendet, der aus seinem Büro die Augmented Reality Anwendung des Monteurs vor Ort steuern kann. Der geschulte Monteur kann dynamisch Informationen in die Anwendung einblenden und somit auf die Probleme des Monteurs vor Ort reagieren.

Neben der Optimierung von Wartungs- und Reparaturvorgängen gibt es auch aktuelle Ansätze, um Prozesse in der laufenden Produktion zu verbessern. Doshi et al. [26] haben 2016 eine Software entwickelt, die beim manuellen Schweißen von Autoteilen den Schweißpunkt visuell hinterlegt. Diese Arbeit baut auf der bereits beschriebenen Arbeit von Zhou et al. [19] auf. Durch bessere Technologien konnten Doshi et al. eine Produktionssteigerung von mindestens 15% nachweisen. Zhang et al. [27] zeigen eine Möglichkeit, dreiachsige CNC-Werkzeugmaschinen im laufenden Betrieb mit Informationen anzureichern. Zwei durchgeführte Studien zeigen, dass die erstellte Applikation für diesen speziellen Anwendungsfall hilfreich ist. Die Tester sind sich jedoch einig, dass die Anwendung mit fünf- oder mehrachsigen Systemen an ihre Grenzen stößt. Torok et al. [28] präsentieren eine Möglichkeit, den Arbeitsprozess eines Koordinatenmessgeräts mit Hilfe von Augmented Reality zu verbessern. Sommer et al. [29] entwickeln ein Framework, in dem Mehrwertdienste auf unterschiedliche Art und Weise den Bediener einer Werkzeugmaschine unterstützen. Im Gegensatz zu den vorherigen Projekten, sollen die Anzeige der Informationen sowie die Steuerung des Frameworks über einen Bildschirm erfolgen, der die Schutzscheibe einer Werkzeugmaschine ersetzt. Durch dieses Vorgehen könnten die Schwachstellen von AR-Brillen und Projektoren umgangen werden. Nee et al. [30] führen einen ungewerteten Vergleich weiterer Augmented Reality Projekte in der Industrie durch.

Für die Erstellung von Augmented Reality Anwendungen für die Produktion gibt es nur wenig Unterstützung durch spezifische Frameworks. Andre [31] hat 2013 ein allgemeines Framework zum Erstellen von Augmented Reality Anwendungen entwickelt, bietet jedoch keine spezifische Unterstützung für industrielle Anwendungen an. Ramirez et al. [32] zeigen lediglich die Notwendigkeit eines Frameworks für Anwendungen mit Produktionsfokus, bieten jedoch keine konkrete Umsetzung an. Kollatsch et al. [33] hingegen präsentieren eine prototypische Lösung eines Frameworks zu Erstellung von industriellen Augmented Reality Anwendungen. Das Framework unterstützt den Benutzer beim Anreichern von virtuellen Objekten mit Daten. Der Fokus dieses Frameworks liegt darin, Rohdaten möglichst schnell in einer simpel designten Oberfläche anzuzeigen. Informationen aus den Daten zu gewinnen und aus Informationen virtuelle Objekte zu erstellen, wird vom Framework nicht unterstützt. Der Mangel an Frameworks zur Entwicklung von Augmented Reality Anwendungen für die Industrie zeigt eine Möglichkeit, den Stand der Technik voran zu treiben.

3.2 Augmented Reality SDKs

In diesem Abschnitt werden Software Development Kits (SDK) für die Entwicklung von Augmented Reality Anwendungen untersucht. Ein AR-SDK bietet grundlegende Methoden an, um virtuelle Elemente in die echte Welt zu übertragen. Die meisten Augmented Reality Frameworks beinhalten ein AR-SDK, um auf dessen Funktionalität aufbauen zu können. Aus diesem Grund werden in diesem Kapitel verschiedene AR-SDKs darauf untersucht, ob sie als potentielle Basis für das AR-Framework in Frage kommen.

Jedes SDK hat unterschiedliche Eigenschaften und Funktionalitäten, von denen nicht alle gleich relevant sind. Um relevante Eigenschaften zu finden, werden im Folgenden drei Anforderungen an ein Augmented Reality SDK nach Amin und Govilkar [34] aufgelistet:

Ein Augmented Reality SDK muss…

- … Marker erkennen und verfolgen können.
- … virtuelle Elemente in der realen Welt platzieren können.
- … mit möglichst vielen Plattformen kompatibel sein.

Diese Anforderungen lassen sich instinktiv nur schlecht quantitativ vergleichen. Als nächster Schritt werden deshalb Metriken definiert, um eine formale Methode zum Vergleich der verschiedenen Technologien zu erhalten. In Tabelle 2 sind erstellte Metriken für jede der drei genannten Eigenschaften abgebildet. Diese Metriken werden an die Anforderungen, an das zu entwickelnde Framework angepasst. Augmented Reality Anwendungen für die Industrie benötigen zum Beispiel kein Position-gebundenes Tracking, da dies in einer Arbeitsumgebung wie einer Werkzeughalle oder innerhalb einer Werkzeugmaschine zu ungenau wäre. In Tabelle 3 sind mögliche Belegungen der drei untersuchten Eigenschaften definiert.

TABELLE 2 - ZUORDNUNG VON SDK EIGENSCHAFTEN UND ENTSPRECHENDER METRIK

Eigenschaft	**Metrik**
Marker Erkennung und Tracking	Anzahl der unterstützten, relevanten Tracking Methoden
Platzierung von virtuellen Elementen	Anzahl der unterstützten, relevanten Darstellungsformen
Plattformkompatibilität	Anzahl der kompatiblen Plattformen

TABELLE 3 - GENAUE DEFINITION DER MÖGLICHEN BELEGUNGEN DER EIGENSCHAFTEN

Eigenschaft	**Relevante Belegungen**
Marker Erkennung und Tracking	2D Tracking, 3D Tracking, Gesichtserkennung
Platzierung von virtuellen Elementen	2D Elemente, 3D Elemente, Animationen, Videos
Plattformkompatibilität	Alle unterstützten Plattformen

Nachfolgend werden sechs, nach Amin und Govilkar [34] und Marneanu et al. [35] aktuelle SDKs für Augmented Reality Anwendungen in alphabetischer Reihenfolge vorgestellt und abschließend durch jeweils eine Tabelle zusammengefasst.

3.2.1 ARMEDIA

ARmedia ist ein Framework, bestehend aus einer Echtzeit-Tracking-Komponente, einer Echtzeit-Rendering-Komponente und einer Schnittstellen-Komponente. Derzeit[4] unterstützt ARmedia lediglich Android und iOS als Zielplattform, hat jedoch angekündigt, dass eine Version für Windows und OS X zeitnah erscheinen wird. Die Erkennung von Markern und Formen funktioniert durch die Nutzung mehrerer Bibliotheken wie OpenCV, OpenNI oder Inglobe Tracker. Durch diese Technologien kann ARmedia 2D- und 3D-Marker erkennen, sowie ein GPS als Tracking Signal verwenden. Durch die potentielle Verwendung von mehreren Grafik-Engines, bietet ARmedia die Möglichkeit statische 2D- und 3D- sowie animierte 3D-Modelle auf Basis des erkannten Markers zu platzieren. ARmedia ist unter

[4] Stand 19.9.2016

einer Closed-Source Lizenz veröffentlicht, bietet jedoch eine kostenlose, rein private Nutzung, sowie eine kostenpflichtige, kommerzielle Nutzung an. Abbildung 11 fasst alle relevanten Eigenschaften von ARmedia zusammen.

ABBILDUNG 11 - AUSWERTUNG VON ARMEDIA AUF BASIS DEFINIERTER METRIKEN

3.2.2 ARTOOLKIT

ARToolKit ist eine Bibliothek für Marker basiertes Tracking, die unter einer Open-Source Lizenz veröffentlicht wurde. ARToolKit unterstützt Android, iOS, Windows Phone, Windows, OS X und Linux mit einem jeweils eigenen SDK. Zusätzlich bietet ARToolKit Skripte für Unity an, die sämtliche Funktionalitäten des SDKs in Unity direkt wiederspiegeln. Zum Erkennen der Marker nutzt ARToolKit die Tracking Bibliothek OpenCV. Als Grafik-Engine fungiert OpenSceneGraph, eine Open-Source Lösung basierend auf OpenGL. Sie ist in der Lage 2D-, 3D-, und animierte 3D-Modelle anzuzeigen. Wenn ARToolKit direkt in Unity genutzt wird, verwenden die Skripte Funktionalitäten aus der Unity Engine und nicht die oben genannten Hilfstechnologien. Die Open-Source Lizenz beinhaltet die kostenlose private und kommerzielle Nutzung. Eine Übersicht der Eigenschaften von ARToolKit kann Abbildung 12 entnommen werden.

ABBILDUNG 12 - AUSWERTUNG VON ARTOOLKIT AUF BASIS DEFINIERTER METRIKEN

3.2.3 D'FUSION

D'Fusion ist ein aus zwei Komponenten bestehendes Softwarepaket, das größtenteils Web basierte Augmented Reality Anwendungen ermöglicht. Für native Android und iOS Anwendungen benutzt

D'Fusion eine eigene Engine namens D'Fusion Augmented Reality. Sowohl 2D-, als auch 3D-Modelle können, wie in Abbildung 13 zu sehen, animiert dargestellt werden. Für die Darstellung von animiertem Inhalt in Web Anwendungen setzt D'Fusion auf Adobe Flash. Das Tracking wird von D'Fusion Computer Vision übernommen. Dieses Tool unterstützt die Erkennung von 2D- und 3D-Modellen sowie eine Gesichtserkennung. Beide Komponenten sind Eigenentwicklungen und unter einer Closed-Source Lizenz veröffentlicht. Mit dem Total Immersion D'Fusion Studio gibt es eine Entwicklungsplattform, die beide Komponenten von D'Fusion beinhaltet und über eine rein grafische Oberfläche anbietet. Das komplette D'Fusion Paket kann in einer kostenlosen privaten Lizenz und in einer kostenpflichtigen gewerblichen Lizenz erworben werden.

ABBILDUNG 13 - AUSWERTUNG VON D'FUSION AUF BASIS DEFINIERTER METRIKEN

3.2.4 METAIO

Das Metaio SDK ist ein in Deutschland entwickeltes modulares Framework zur Erstellung von AR Anwendungen, das im Mai 2015 von Apple aufgekauft wurde. Seit Dezember 2015 ist es nicht mehr möglich, die Software zu erwerben oder zu nutzen. Metaio wird nur der Vollständigkeit halber erwähnt und wird nicht durch die festgelegten Metriken geprüft.

3.2.5 VUFORIA

Vuforia bietet SDKs zur einfachen Erstellung von Augmented Reality Anwendungen an. Das Unternehmen wurde im Oktober 2015 von PTC gekauft. PTC [36] hat jedoch angekündigt, im Gegensatz zu Metaio, ihr Produkt weiterhin auf dem Markt zu halten. Vuforia hat sich auf den mobilen Markt für Android und iOS Systeme spezialisiert, bietet aber auch ein SDK für die Universal Windows Plattform an und unterstützt somit Windows 10 Anwendungen. Neben 2D-Markern, erkennt Vuforia 3D-Modelle, bietet jedoch keine Unterstützung für GPS oder Gesichtserkennung. Statischer und animierter 2D- sowie 3D-Inhalt kann über OpenGL in den erstellen Anwendungen angezeigt werden. Vuforia bietet eine eingeschränkte kostenlose Nutzung für private Zwecke an. Für den vollen Umfang der Technologie müssen sowohl private, als auch gewerbliche Nutzer eine kostenpflichtige Lizenz erwerben. Abbildung 14 fasst die Merkmale von Vuforia zusammen.

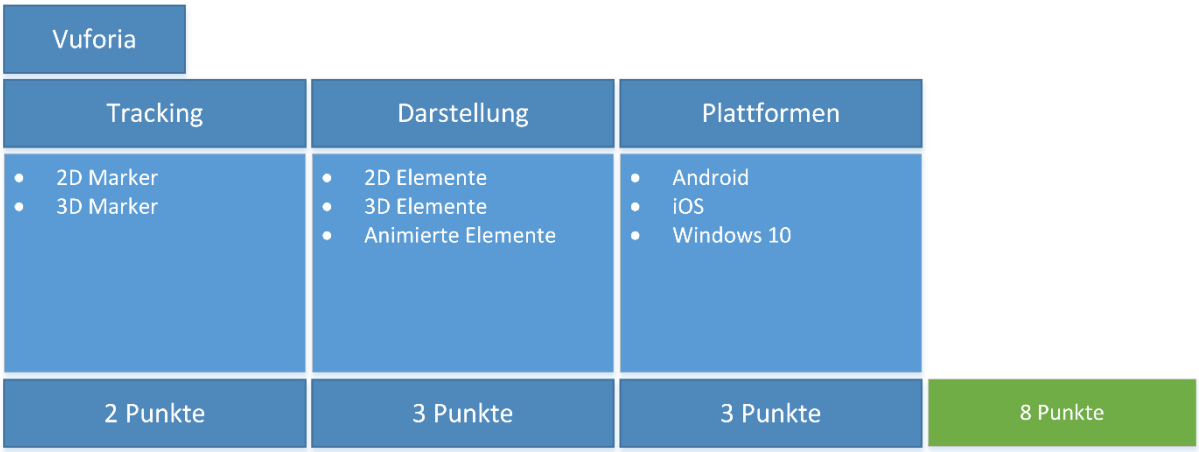

ABBILDUNG 14 - AUSWERTUNG VON VUFORIA AUF BASIS DEFINIERTER METRIKEN

3.2.6 WIKITUDE

Wikitude ist ein SDK zum Erstellen von mobilen Augmented Reality Anwendungen für Smartphones und Smartglasses. Hierfür bietet Wikitude neben einem SDK mit der Wikitude App eine Möglichkeit, erstellte AR Anwendungen zu publizieren. Zielplattformen sind Android und iOS Geräte sowie Smartglasses von Google, Epson, Vuzix und ODG. Für die Erkennung von Markern wird eine eigens entwickelte Technologie genutzt. Diese erlaubt das Tracking von 2D-Markern in Kombination mit einer Standorterkennung durch GPS. Wie in Abbildung 15 zu entnehmen ist, fehlt die Unterstützung von 3D-Markererkennung. Zur Erweiterung der realen Welt bietet Wikitude die Möglichkeit, animierte 2D- sowie 3D-Modelle und HTML5-basierte Videos anzuzeigen. Wikitude bietet ein kostenloses und ein kostenpflichtiges Nutzungsmodell an. Im kostenlosen Modell sind nicht alle Funktionen der kostenpflichtigen Version enthalten. Wird Wikitude gewerblich benutzt, steht keine kostenlose Variante zur Verfügung.

ABBILDUNG 15 - AUSWERTUNG VON WIKITUDE AUF BASIS DEFINIERTER METRIKEN

3.2.7 ERGEBNISANALYSE DES VERGLEICHS

Durch die Nutzung von einheitlichen, auf einem Punktesystem basierenden Metriken, ist die Auswertung der Analyse vom Stand der Technik einfach zu bewältigen. Sowohl ARToolKit, als auch Wikitude heben sich in den relevanten Eigenschaften deutlich von den anderen Technologien ab. Um zu entscheiden, ob eine der beiden Technologien dafür geeignet ist, als Basis für das AR-Framework dieser Arbeit zu dienen, werden in Abbildung 16 zwei Augmented Reality spezifische Anforderungen

aus der Spezifikation mit den Eigenschaften der beiden Technologien abgeglichen. Wikitude besitzt die Möglichkeit, die API der Unity Engine anzusprechen, bietet aber nicht wie ARToolKit die Möglichkeit, alle Funktionalitäten des SDKs direkt in der Entwicklungsumgebung von Unity zu nutzen. Darunter fällt die explizit genannte Einrichtung der virtuellen Welt in Unity. Die geforderte Unterstützung von verschiedenen PC Betriebssystemen mit angeschlossener AR-Brille kann Wikitude im Vergleich zu ARToolKit nicht erfüllen.

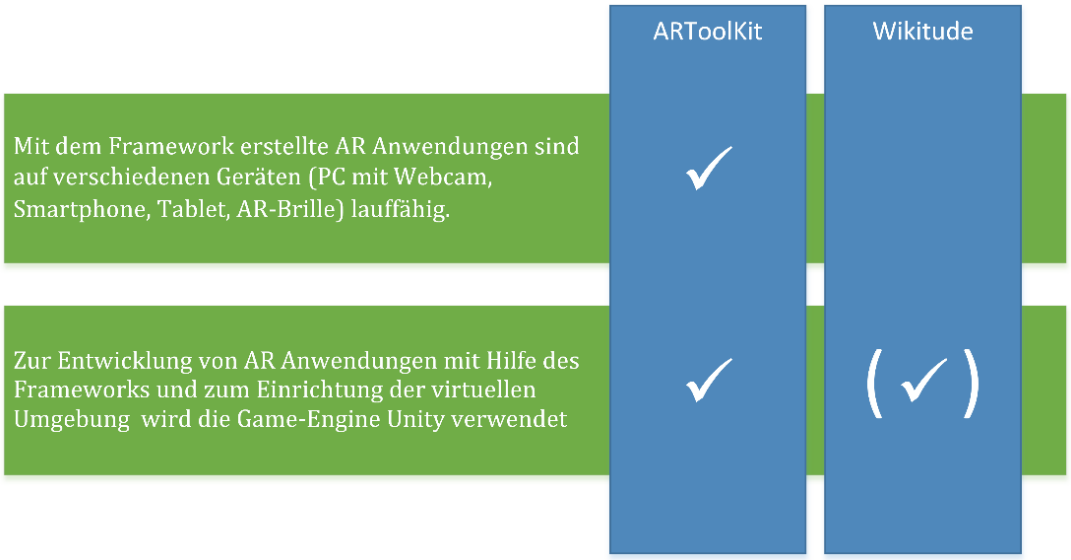

ABBILDUNG 16 - VERGLEICH VON ANFORDERUNGEN UND SDK FUNKTIONALITÄTEN

Die Analyse vom Stand der Technik im Bereich Augmented Reality SDKs liefert somit ein eindeutiges Ergebnis. ARToolKit bietet alle Funktionalitäten, die für den Augmented Reality Teil des AR-Frameworks nötig sind und wird deshalb für diese Komponente genutzt.

3.3 VERWALTUNG VON PRODUKTIONSRELEVANTEN DATEN

Das Erfassen und Auswerten von Daten bildet das Fundament der digitalen Produktion. Aus diesem Grund müssen im AR-Framework verschiedene Arten von Daten erfasst und verarbeitet werden können. Um zu analysieren wie der Stand der Technik beim Beziehen und Verwalten von Daten in der Produktion aussieht, muss zuerst definiert werden, welche Datentypen für diese Arbeit relevant sind. Hierfür werden die Anforderungen aus der Spezifikation zu Hilfe genommen. PFA13 aus den funktionalen Anforderungen definiert folgende, alphabetisch sortierte Daten als relevant:

- Dokumente (Handbücher, Anleitungen)
- Geometrieinformationen (CAD)
- Maschinendaten (OPC UA, ADS)
- Planungs- und Steuerdaten (ERP, MES)
- Schaltpläne (EPLAN)
- Steuerungsdaten (NC-Programme)

Nachfolgend wird der Stand der Technik zum programmatischen Zugriff und der Verwaltung dieser Daten dargelegt. Eine Unterscheidung zwischen den verschiedenen Datentypen sorgt für eine bessere Übersicht. Wichtig hierbei ist der Fokus auf den programmatischen Zugriff, da dieser relevant dafür ist, die Daten zu analysieren und für eine Augmented Reality Lösung aufzubereiten.

3.3.1 DOKUMENTE

Um textbasierte Dokumente maschinell zu verarbeiten, können mehrere Techniken genutzt werden. Meaningcloud, eine Cloud basierte Software as a Service (SaaS) Lösung, setzt auf eine Kombination aus Computerlinguistik, um vorhandenen Text zu deuten und maschinellem Lernen, um gedeutete Texte zu analysieren. Der Fokus von Meaningcloud liegt auf der Erkennung von Themengebieten innerhalb eines Dokuments und dem Verknüpfen mehrerer Dokumente mit ähnlichem Inhalt. Dieser Service bietet ein SDK für viele gängige Programmiersprachen, wie Java, Python, PHP und Visual Basic. Aylien ist ein weiteres Werkzeug, um textuelle Dokumente zu analysieren. Der Inhalt von Dokumenten wird von Aylien gedeutet und Schlüsselwörter und Informationen können in Form von JSON-Dateien extrahiert werden. Die aktuelle Forschung zur Verarbeitung textueller Dokumente befasst sich zu einem großen Teil mit Sentimentanalysen [37]. Dieser Analysetyp versucht durch maschinelles Lernen, zum Beispiel mit dem „Multinomial Naive Bayes“ [38] und dem „Support Vector Machine“ [39] Algorithmus, die Stimmung in einem Text zu erkennen. Panichella [40] versucht in einer im November 2015 veröffentlichten Arbeit, Inhalte von E-Mails durch textuelle Analyse zu erfassen. Zu seinem Ansatz hat Panichella jedoch noch keine Ergebnisse veröffentlicht. Um tatsächlich den Inhalt eines Textes maschinell zu erfassen und aufzubereiten, bedarf es momentan Konventionen innerhalb des Textes. Derzeit wird am Institut für Steuerungstechnik der Universität Stuttgart ein Modell für Industrie 4.0 konforme Dokumente entwickelt [41]. Dieses Modell könnte es durch definierte Strukturen ermöglichen, aus Dokumenten maschinell brauchbare Informationen zu gewinnen.

3.3.2 GEOMETRIEINFORMATIONEN

Computer Aided Design (CAD)-Modelle sind in der Steuerungstechnik eine beliebte Möglichkeit, Bauteile geometrisch sowie physikalisch zu beschreiben. Ein CAD-Modell kann in unterschiedlichen Datenformaten beschrieben werden. CAD.Net ist eine Bibliothek für das .NET Framework, mit der mehrere CAD-Formate eingelesen werden können. Nach dem Einlesen bietet die Bibliothek Zugriff auf die Daten und zusätzlich die Möglichkeit, die in CAD beschriebenen Teile grafisch darzustellen. Im Juli 2016 wurde von Telenta et. Al. [42] eine Bibliothek zur Kombinierung von CAD-Daten und Daten die physikalische Vorgänge beschreiben, entwickelt. Diese Bibliothek bietet Zugriff auf die Daten sowie die Möglichkeit das Ergebnis zu visualisieren.

3.3.3 PLANUNGSDATEN

In diesem Unterkapitel beschriebene Systeme befassen sich mit ERP und MES Systemen und somit mit dem oberen Ende der Automatisierungspyramide. Je nach genutzter Software bieten die Systeme einen Export von Daten in verschiedenen Formaten an. Der ERP Data Import Export Wizard bietet eine Konvertierung von exportierten Datensätzen aus ERP Systemen wie SAP ERP oder Microsoft ERP in verschiedene offene Formate an. Der Benutzer kann zwischen HTML, CSV, RTF sowie Word und Excel als Zielformat wählen. Diese Formate können anschließend über Standardbibliotheken höherer Programmiersprachen maschinell eingelesen und verarbeitet werden. Einige ERP und MES System wie beispielsweise SAP ERP und SAP MES bieten Bibliotheken für höhere Programmiersprachen an, um direkt auf Daten innerhalb des Systems zugreifen zu können. Daffner und Jürgensen [43] haben den SAP .Net Connector genutzt um ERP-Daten an ein Web Frontend zu übermitteln. Winkler et al. [44] beschreiben Software, die MES Daten importieren kann und für Schulungszwecke zur Verfügung stellt.

3.3.4 SCHALTPLÄNE

Wie in den Anforderungen beschrieben, soll das Framework eine Unterstützung für digitale Schaltpläne aus EPLAN Electric P8 bereitstellen. Diese Schaltpläne liegen laut einem persönlichen

Gespräch mit einem Vertreter von EPLAN P8 ausschließlich verschlüsselt dar. EPLAN Electric P8 bietet eine Exportfunktion, um Informationen von erstellten Schaltplänen teilweise in unverschlüsselte Datenformate zu übertragen. Shojaei et al. [45] haben diese exportierbaren Daten eingelesen und als Grundlage genutzt, um eine 3D Visualisierung der Schaltpläne zu erstellen. Die exportierten Daten aus EPLAN Electric P8 liegen als Excel Tabellen vor und können so über frei zugängliche Bibliotheken eingelesen werden.

3.3.5 STEUERUNGSDATEN

Steuerungsdaten liegen unter anderem in Form von Code für NC- und SPS-Systeme vor. Der Quellcode beschreibt das Verhalten der Steuerung. Durch den definierten Aufbau der Programmiersprache für NC-Systeme gibt es mehrere Versuche mit Hilfe von Open Source Technologien G-Code[5] maschinell einzulesen. Lee et al. [46] lesen NC-Programme ein, um virtuelle Abläufe des NC-Programms zu simulieren. Laut der internationalen Norm IEC 61131 [47] gibt es für SPS-Programme fünf verschiedene Programmiersprachen. Darvas, Majzik und Vinuela [48] führten 2016 eine Studie durch, in der sie SPS-Programme eingelesen, analysiert und modellbasiert auf ihre Korrektheit überprüft haben.

3.3.6 MASCHINENDATEN

Ein wichtiger Teil der zu analysierenden Daten kommt direkt aus der Maschine. Das OPC UA Protokoll der OPC Foundation [49] bietet eine standardisierte Möglichkeit, Daten von der Steuerung abzugreifen und Daten an die Steuerung zu senden. Für die Entwicklung eigener auf OPC UA basierender Lösungen bietet die OPC Foundation SDKs für mehrere Programmiersprachen an [50]. Auf Basis dieser SDKs gibt es öffentlich zugängliche Frameworks wie das Unified Architecture Framework, die bei der Erstellung einer OPC UA Applikation unterstützen. Laut einem Artikel von H. Junker [51] im Oktober 2015 wird OPC UA von den meisten Herstellern für die Industrie unterstützt.

3.4 ERKENNTNISSE

Für die beiden, für dieses Projekt relevanten Themengebiete wurde jeweils eine Analyse des Stands der Technik durchgeführt.

Eine Analyse von aktuellen Augmented Reality Anwendungen in der Produktion zeigt, dass die Relevanz dieser Anwendungen weiter zunimmt. Gleichzeitig besteht ein Mangel an unterstützenden Technologien zum Erstellen fachspezifischer Anwendungen. Ein Framework speziell für industrienahe Anwendungen kann den Stand der Technik weiter voranbringen. Das ARToolKit besitzt alle nötigen Eigenschaften, um die Basis eines solchen Frameworks zu bilden. Aus diesem Grund wird das ARToolKit in dieser Arbeit für den Augmented Reality Teil des Frameworks verwendet. Die Entwicklung eines eigenen AR-SDKs ist nicht nötig.

Bei dem Zugriff und der Verwaltung von verschiedenen Daten bietet der Stand der Technik sehr spezifische Lösungen für jedes Problem. Kein Framework aus dem in Kapitel 3.3 präsentierten Stand der Technik kann mehrere der genannten Datentypen verwalten. Für die Verwaltung einzelner Daten gibt es jedoch jeweils eine oder mehrere Lösungen. Daraus folgt, dass eine Eigenentwicklung im Bereich der Datenhaltung nötig ist, um die geforderte Funktionalität des AR-Frameworks zu ermöglichen.

Aus diesen gewonnenen Informationen gehen zwei Gebiete hervor, auf denen diese Arbeit den Stand der Technik erweitern kann. Eine Sammlung, die Zugriff auf verschiedene produktionsrelevante

[5] Maschinensprache zur Steuerung von CNC Maschinen

Datenquellen bietet, in dem sie mehrere der betrachteten Lösungen kombiniert, trägt in kleinem Umfang zu einer Neuerung im Stand der Technik bei. Der zweite, weitaus größere Beitrag ist der Bau einer Brücke zwischen der Datenhaltungs- und der Augmented Reality Komponente. Beide Komponenten sind im aktuellen Stand der Technik nahezu unverbunden. Die Möglichkeit für einen Entwickler von Augmented Reality Anwendungen für den Industriemarkt, direkt auf verschiedene relevante Daten zuzugreifen, kann den Erstellungsprozess vereinfachen. Wenn diese Daten zusätzlich noch für den Einsatz in einer Augmented Reality Anwendung präpariert sind, entsteht ein weiterer Mehrwert für den Entwickler. Diese Neuerung führt zu einer einfacheren Rückführung der Daten in der digitalen Produktion.

4 KONZEPTION

Nachdem die theoretischen Grundlagen und der Stand der Technik beschrieben wurden, behandelt dieses Kapitel die konkrete Konzeption des zu erstellenden Frameworks. Ein abstrakter Überblick fast zusammen, was das AR-Framework an Funktionalität bieten soll. Darauffolgend wird der Aufbau des AR-Frameworks auf Komponentenlevel beschrieben. Der Sinn hinter den verschiedenen Komponenten steht in diesem Kapitel im Vordergrund. Die Umsetzung der Komponenten wird in Kapitel 5 detailliert behandelt. Durch diese Struktur wird das AR-Framework Schritt für Schritt detaillierter dargestellt.

4.1 GESAMTÜBERBLICK

Ein Überblick über das Framework fasst sämtliche Funktionalitäten des Frameworks zusammen und erklärt somit den Anwendungszweck des Frameworks. Für eine strukturierte Übersicht, bedient sich dieses Kapitel an dem in Kapitel 1.2 definierten Orientierungssatz. Dieser Satz beschreibt die Art und Weise, wie das Framework benutzt werden soll.

Orientierung:

Ein Entwickler wird von einem Framework durch den kompletten Zyklus von der **Erfassung verschiedener Daten**, über die **Datenanalyse und Mehrwertgewinnung** bis hin zur **Rückführung der Daten** in die Produktion begleitet.

Die fett gedruckten Teile des Satzes beschreiben die drei Kernfunktionalitäten des Frameworks. Für jede dieser Funktionalitäten wird nachfolgend beschrieben, wie das Framework diese abstrakten Anforderungen umsetzt.

4.1.1 ERFASSUNG VERSCHIEDENER DATEN

Um Daten erfassen zu können, bietet das Framework einem Entwickler die Möglichkeit, verschiedene Daten durch ein einheitliches Vorgehen zu beziehen. Die im Stand der Technik beschriebenen Technologien nutzen unterschiedliche Schnittstellen, deren Bedienung sich stark unterscheidet. Ein Entwickler muss sich entsprechend in jede Technologie einarbeiten, um diese nutzen zu können. Das AR-Framework umgeht diese Hürde, indem es eine zusätzliche Abstraktionsschicht über die Technologien legt und somit eine Schnittstelle anbietet, die ein einheitliches Konzept für die Erfassung von Daten anbietet. Abbildung 17 zeigt eine abstrakte Darstellung dieser zusätzlichen Schicht. Die drei beispielhaften Technologien bieten jeweils unterschiedliche Schnittstellen für den Zugriff auf Daten. Die Framework Wrapper vereinheitlichen diesen Zugriff.

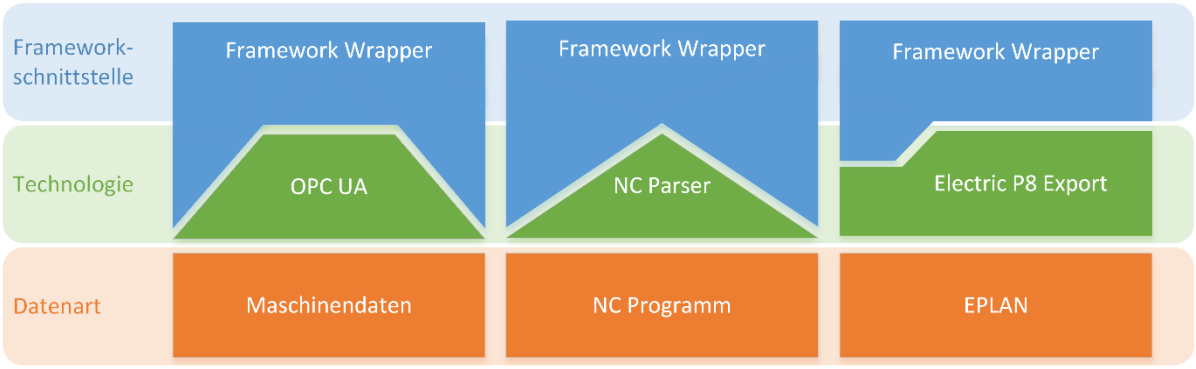

ABBILDUNG 17 - BEISPIELHAFTE DARSTELLUNG VON EINHEITLICHEM ZUGRIFF AUF VERSCHIEDENE DATEN

Der in Abbildung 17 bereits angedeutete modulare Aufbau des Technologiewrapper-Konzepts bietet dem Entwickler weitere Vorteile. Einerseits verwaltet das Framework die genutzten Technologien, sodass sich der Entwickler nur um die Aktualität des Frameworks kümmern muss. Auf der anderen Seite bietet ein modularer Aufbau die Möglichkeit, weitere Technologien mit minimalem Aufwand einzubinden. Für diesen Zweck stellt das Framework gemäß den Framework Guidelines von Cwalina und Abrams [11] Visual Studio Symbolvorlagen bereit.

Zur Erfassung von Daten bietet das Framework zwei methodisch unterschiedliche Möglichkeiten an. Daten können entweder einmalig geladen oder über einen ständigen Datenfluss aktuell gehalten werden. Für NC-Programme oder einen EPLAN reicht zum Beispiel eine einmalige Erfassung aus. Für Maschinendaten, wie zum Beispiel die derzeitige Achsposition müssen Daten ständig aktualisiert werden. Für beide Arten der Datenerfassung bietet das Framework Hilfestellungen durch die bereits genannten Symbolvorlagen. Sind die Daten geladen oder ein Datenstrom hergestellt, bietet das Framework diese Daten für zwei verschiedene Zwecke an. Zum einen kann der Entwickler auf die Daten zugreifen und sie direkt verwenden. Zum anderen werden die Daten für Mehrwertdienste aufbereitet.

4.1.2 Datenanalyse und Mehrwertgewinnung

Dieser Teil des Frameworks verarbeitet die bezogenen Daten, um aus ihnen Informationen zu gewinnen. Die gewonnenen Informationen werden für die Nutzung in Unity vorbereitet. Wie in Abbildung 18 beschrieben, kann eine Analyse nicht nur auf Basis einer Datenart durchgeführt werden. Auch die Kombination mehrerer Daten von unterschiedlichen Typen wird vom Framework unterstützt. Anhand der im Beispiel gewählten Datenarten ist ebenfalls zu erkennen, dass einmalig geladene, konstante Daten (NC Daten) und sich dynamisch ändernde Daten (OPC UA Daten) für eine Analyse kombiniert werden können. Die Analyse produziert Informationen, die auf zwei unterschiedliche Arten genutzt werden können. Das Ergebnis kann über eine Schnittstelle direkt für die Nutzung innerhalb von Unity Skripten angeboten werden. Die zweite Möglichkeit ist die Nutzung der Ergebnisse für eine weitere Analyse.

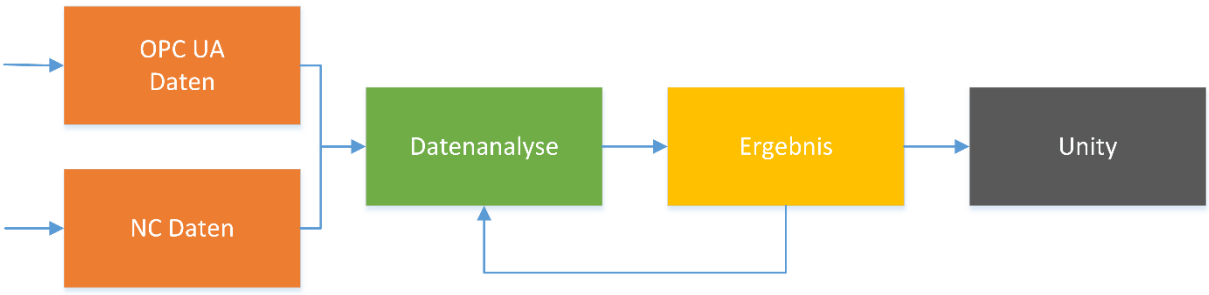

Abbildung 18 - Ablauf der Analyse von Daten innerhalb des AR-Frameworks

Im Beispiel aus Abbildung 18 könnte eine Analyse der Daten dazu führen, dass ein Datensatz entsteht, der aktuelle Maschinendaten zum Zeitpunkt einer bestimmten Zeile NC-Code bereitstellt. Diese Daten könnten in Unity genutzt werden, um beim Fräsvorgang einzublenden, welche Zeile Code gerade ausgeführt wird. Andererseits könnte man diese Information noch mit Spezifikationsdaten der Fräsmaschine kombinieren, um zu erfahren, welche Zeile NC-Code laut Spezifikation sicherheitskritische Werte in der Maschine produziert. Da das Framework die Rückführung von Ergebnissen in die Analysephase ermöglicht, wären beide Anwendungsideen durch das Framework realisierbar.

4.1.3 Rückführung der Daten

In Abbildung 18 wird gezeigt, dass Unity bei der Rückführung der gewonnenen Daten in die reale Produktion eine Rolle spielt. Abbildung 19 beschreibt diese Rolle im Detail. Unity Skripte sind in der Lage, Analyseergebnisse aus dem AR-Framework zu beziehen, indem sie die bereitgestellten Schnittstellen nutzen. Das Framework kann durch die Einbindung der *UnityEngine.DLL*-Bibliothek Ergebnisse bereits Unity konform vorbereiten und übergeben. Diese Informationen werden innerhalb des Skriptes an die Unity Engine der Unity Entwicklungsumgebung weitergeleitet. Hierbei können innerhalb des Skripts grafische Elemente erstellt und mit Hilfe der Informationen befüllt und positioniert werden. Für die AR-Fähigkeit sorgen Skripte aus dem ARToolKit. Eine Kombination beider Skripttypen lässt die Unity Umgebung letztendlich eine Augmented Reality Anwendung erstellen.

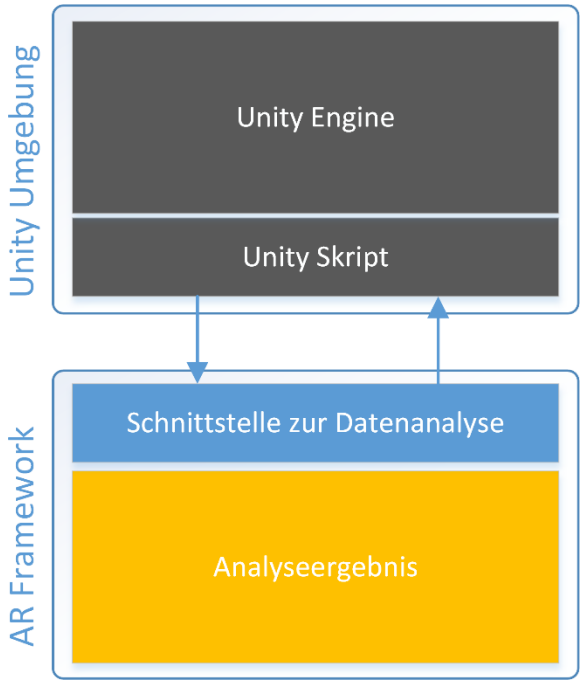

Abbildung 19 - Rückführung der Analyseergebnisse in die reale Produktion mit Hilfe des AR-Frameworks

4.1.4 Zusammenfassung

In Abbildung 20 sind alle in diesem Kapitel beschriebenen Abläufe und somit das grundlegende Konzept des AR-Frameworks zusammengefasst. Das AR-Framework sammelt verschiedene Daten durch die Nutzung unterschiedlicher Technologien. Sind die Daten im AR-Framework vorhanden, können Analysen ausgeführt werden, um mehrwertige Informationen aus den Rohdaten zu gewinnen. Anschließend werden diese Informationen an Unity weitergeleitet. Dort wird mit Hilfe des ARToolKits eine Augmented Reality Anwendung konstruiert. Diese Anwendung kann in der realen Produktion eingesetzt werden, um von den erfassten Daten zu profitieren.

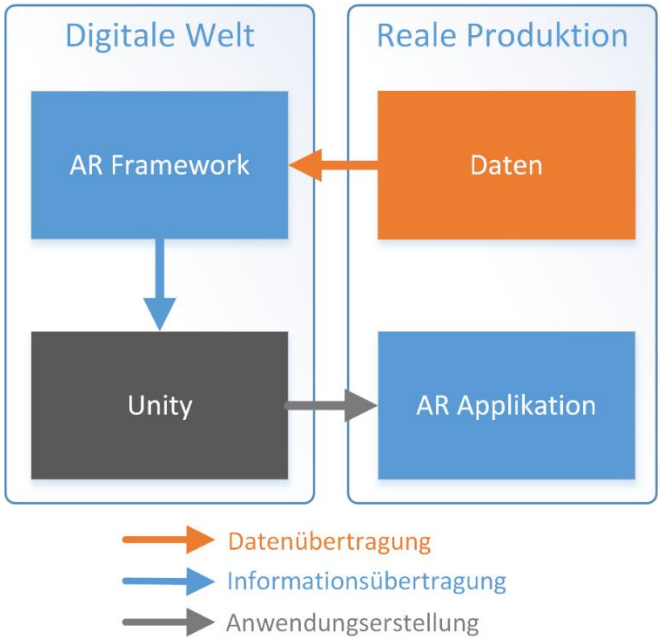

ABBILDUNG 20 - KONZEPTIONELLER ÜBERBLICK ÜBER DAS ZU ERSTELLENDE FRAMEWORK

4.2 AUFBAU

In diesem Kapitel wird die Umsetzung des Konzepts für das AR-Framework näher betrachtet. Nach einer mehr funktionalen Beschreibung in Kapitel 4.1 wird in Abbildung 21 das Framework eine Stufe detaillierter dargestellt. Das AR-Framework besteht aus vier verschiedenen internen Komponenten, die aufeinander zugreifen. Jede dieser Komponenten wird nachfolgend beschrieben.

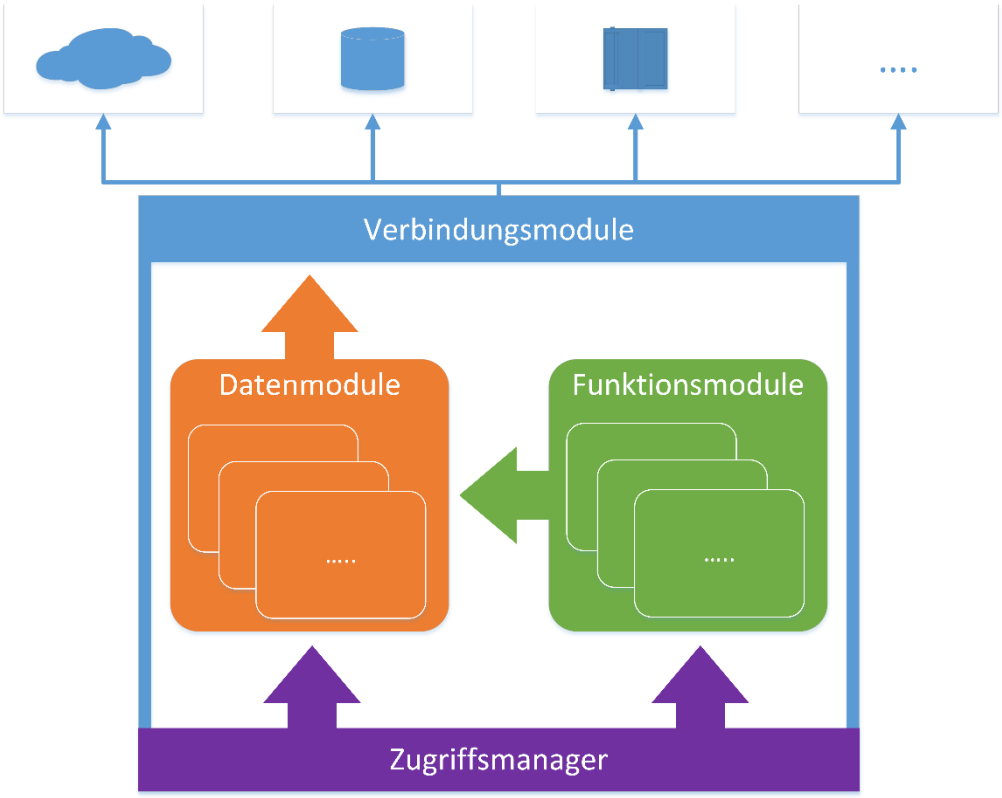

ABBILDUNG 21 - ABSTRAKTER INTERNER AUFBAU DES AR-FRAMEWORKS

4.2.1 DATENMODUL

Ein Datenmodul beinhaltet semantisch zusammenhängende Daten, die Möglichkeit diese Daten zu laden und eine definierte Schnittstelle, um auf die enthaltenen Daten zuzugreifen.

Diese Komponente spiegelt Daten wieder, die aus der Produktion ausgelesen werden. Für jeden Datentyp gibt es ein Datenmodul. Da unterschiedliche Daten vom gleichen Typ gleichzeitig vorkommen können, bietet das AR-Framework mehrere Instanzen eines Datenmoduls an. Für den Zugriff auf die verschiedenen Module existiert ein Identifikationsobjekt für jedes Modul. Dieses Objekt ist ein Tupel, bestehend aus dem Typ des Datenmoduls, sowie einer einzigartigen Nummer. Die Vergabe und die Verwaltung des Identifikationsobjekts übernimmt das AR-Framework. Möchte ein Entwickler auf ein Datenmodul zugreifen, erhält dieser mit Hilfe des Identifikationsobjekts eine Referenz auf das Modul. Die Modulschnittstelle, mit deren Hilfe Daten aus dem Modul bezogen werden, legt der Entwickler des Moduls fest. Um eine einheitliche Nutzung zu gewährleisten, definiert das Framework eine Basisschnittstelle, die für jedes Datenmodul zu implementieren ist. Über diese Schnittstelle sind Standardfunktionalitäten wie das Starten und Stoppen des Moduls sowie das Laden von Daten zugänglich. Modulspezifische Funktionalität kann nicht einheitlich vorgegeben werden. Das AR-Framework gibt hierfür jedoch einige Richtlinien vor, um die erweiterten Schnittstellen der Datenmodule möglichst ähnlich zu halten. Wie in Abbildung 21 zu sehen ist, werden Datenmodule von Funktionsmodulen genutzt. Funktionsmodule greifen hierbei über die gleiche Schnittstelle auf die Datenmodule zu, wie ein Entwickler innerhalb seines Skripts. Diese Entscheidung wurde getroffen, um Datenmodule übersichtlicher zu halten und den Wartungsaufwand auf eine Schnittstelle zu beschränken. Um Daten an Funktionsmodule anbieten zu können, muss ein Datenmodul diese Daten zuerst beziehen. Dieser Vorgang geschieht über Verbindungsmodule. Ein Datenmodul kann eine oder mehrere Verbindungsmodule verwenden. Bevor ein Datenmodul gestartet werden kann, muss mindestens ein Verbindungsmodul hinterlegt sein. Bei der Entwicklung eines Datenmoduls wird die Entscheidung getroffen, welche Verbindungsmodule unterstützt werden. Somit ist das Datenmodul dafür zuständig, empfangene Daten aus den Verbindungsmodulen zu speichern und zu verwalten.

Der genaue Lebenszyklus eines Datenmoduls ist, wie in Abbildung 22 zu erkennen, auf zwei Zustände beschränkt. Nach der Initialisierung befindet sich das Datenmodul im inaktiven Zustand. In diesem Zustand wartet das Modul auf die Zuweisung eines Verbindungsmoduls. Nachdem definiert ist, über welches Verbindungsmodul Daten geladen werden können, kann das Modul in den aktiven Zustand wechseln. Beim Übergang wird die Datenverbindung aufgebaut. Ist der Aufbau erfolgreich, wechselt das Modul in den aktiven Zustand. Sollte beim Aufbau der Datenverbindung ein Problem entstehen, bleibt das Modul im inaktiven Zustand und gibt einen Fehler zurück. Während ein Modul im aktiven Zustand ist, kann über die Schnittstelle auf die Daten zugegriffen werden. Wird ein Datenmodul nicht mehr benötigt, kann es heruntergefahren werden. Dafür muss das Modul jedoch zuerst in den inaktiven Zustand überführt werden. In diesem Zustand angekommen, behält das Datenmodul seine zugewiesenen Datenverbindungen und ist jederzeit wieder bereit gestartet zu werden. Das Herunterfahren des Moduls entfernt das Verbindungsmodul und gibt das Datenmodul-Objekt frei. Um das Datenmodul nach dieser Aktion erneut nutzen zu können, muss ein komplett neues Datenmodul vom gleichen Typ angelegt werden.

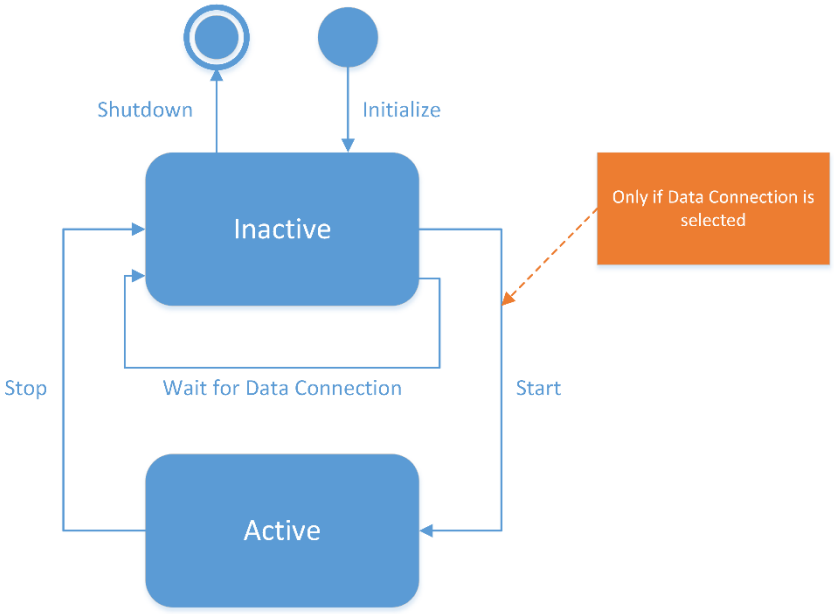

ABBILDUNG 22 - ZUSTANDSGRAPH EINES DATENMODULS IM AR-FRAMEWORK

4.2.2 VERBINDUNGSMODUL

Ein Verbindungsmodul bietet einem Datenmodul Zugriff auf Daten einer externen Datenquelle.

Um Datenmodule mit Daten zu versorgen, ist eine Schnittstelle zur Außenwelt nötig. Wie bereits beschrieben, gibt es unterschiedliche Technologien, um verschiedene Daten zu erfassen, jedoch ist jede dieser Technologien sehr spezifisch zu bedienen. Eine einheitliche Nutzung dieser Technologien kann den Einarbeitungsaufwand für einzelne Technologien minimieren. Verbindungsmodule bieten diese Möglichkeit. Ein Verbindungsmodul ist ein Adapter, der jeweils eine Technologie umschließt und versucht, die Schnittstellen dieser Technologie zu normieren. Das Ziel besteht darin, den Zugriff auf die Technologie so zu vereinfachen, dass der Frameworknutzer letztendlich über einige Grundfunktionalitäten die gewünschten Daten lesen kann. Analog zum Datenmodul kann ein Verbindungsmodul nach dem Initialisieren gestartet werden. Für die Verbindung zur Datenquelle nötige Informationen werden beim Initialisieren übergeben. Mögliche Parameter sind zum Beispiel eine URL zur Datenquelle oder Einstellungsoptionen zur unterliegenden Technologie. Ein Verbindungsmodul kann Daten einmalig laden oder fortlaufend überschreiben. Beide Aktionen werden mit dem Starten des Moduls ausgeführt. Abhängig davon, ob das Ergebnis der Datenbeschaffung positiv oder negativ ist, besitzt das Verbindungsmodul nach dem Start intern einen abrufbaren Datensatz oder nicht. Die verschiedenen Zustände eines Verbindungsmoduls können Abbildung 23 entnommen werden. Sollte kein Laden der Daten möglich sein, kann das Modul gestoppt oder der Lesevorgang wiederholt werden.

Erhaltene Daten werden innerhalb eines Verbindungsmoduls nicht manipuliert oder analysiert. Diese Komponente dient rein der Beschaffung der Daten. Die unbearbeiteten Daten werden vollständig an das übergeordnete Datenmodul weitergeleitet. Dort werden sie bearbeitet und zur Nutzung innerhalb des AR-Frameworks vorbereitet. Bei einem optimalen Verbindungsmodul beschränken sich die auszuführenden Methoden auf das Starten des Moduls sowie dem Methodenaufruf zum Erhalten der Daten.

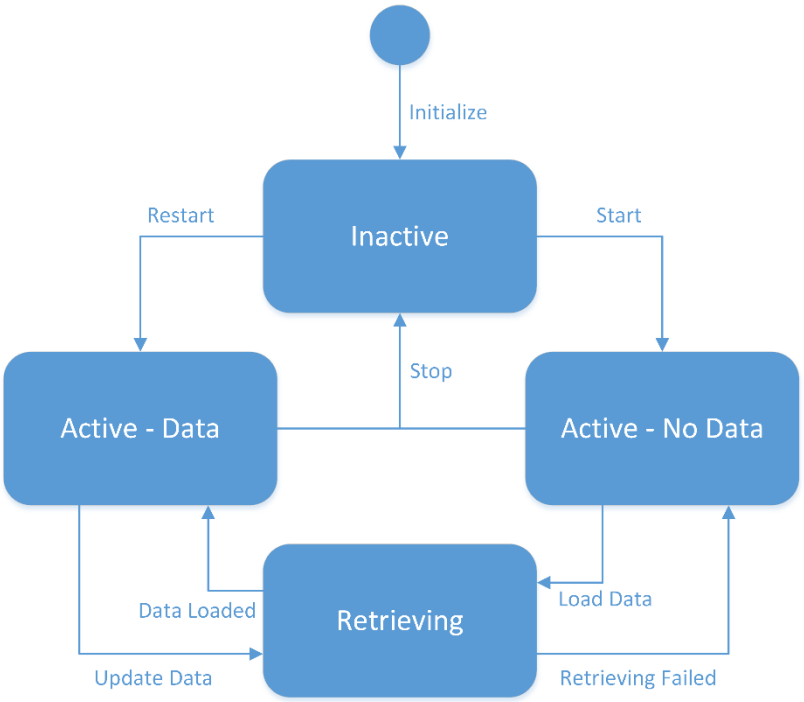

ABBILDUNG 23 - ZUSTANDSGRAPH EINES VERBINDUNGSMODULS IM AR-FRAMEWORK

Das Format dieser Daten wird vom Verbindungsmodulentwickler vorgegeben und muss dem Entwickler eines Datenmoduls bekannt sein. Liest ein Verbindungsmodul Daten nicht nur einmalig, sondern fortlaufend, so bietet das Modul ein Event an, auf das ein Datenmodul reagieren kann, um Änderungen an den Daten zu erhalten. Zusätzlich ist es jedoch jederzeit möglich, einen aktuellen Stand der Daten abzufragen.

4.2.3 FUNKTIONSMODUL

Ein Funktionsmodul bündelt semantisch zusammenhängende Mehrwertfunktionen über ein oder mehrere Datenmodule und produziert Informationen für die Erstellung von Augmented Reality Anwendungen.

Nachdem ein Datenmodul mit Hilfe eines Verbindungsmoduls Daten gesammelt hat, müssen diese Daten ausgewertet werden, um einen Mehrwert zu produzieren. Dies ist die Aufgabe von Funktionsmodulen. Ein Funktionsmodul greift auf ein oder mehrere Datenmodule zu und nutzt die angebotenen Daten, um Informationen zu gewinnen. Im Unterschied zu einem Datenmodul existiert ein Funktionsmodul nur einmal im Framework. Um Zugriff auf dieses statische Funktionsmodul zu erhalten, übergibt der Entwickler alle notwendigen Datenmodule in Form ihrer Identifikationsobjekte. Innerhalb des Frameworks wird das Funktionsmodul mit den Datenmodulen verbunden und eine Referenz des Funktionsmoduls wird an den Entwickler übergeben. Sollten die übergebenen Datenmodule nicht in einem aktiven Zustand sein, reagiert das Framework mit einer Fehlermeldung und gibt keine Referenz auf das Funktionsmodul zurück. Der Ersteller eines Funktionsmoduls definiert, welche Datenmodule für den Gebrauch des Funktionsmoduls nötig sind. Analog zu Daten- und Verbindungsmodulen gibt das Framework ein Grundgerüst der zu implementierenden Methoden für ein Funktionsmodul an. Diese Methoden beschränken sich auf grundlegende Funktionalitäten wie den Zugriff auf Abhängigkeiten zu Datenmodulen oder das Ändern der benutzten Datenmodule. Methoden, die Zugriff auf die Funktionen zur Generierung von Informationen bieten, sind vom

Entwickler frei definierbar. Um eine einfache Nutzung zu unterstützten, gibt es jedoch syntaktische Vorschläge deren Einhaltung das Framework konsistent hält.

In Abbildung 24 ist zu sehen, dass sich die Zustände eines Funktionsmoduls denen eines Datenmoduls ähneln, jedoch nicht identisch sind. Das Verhalten beim Starten und Stoppen eines Funktionsmoduls funktioniert auf die gleiche Art und Weise wie bei einem Datenmodul. Bevor ein Modul heruntergefahren werden kann, muss es gestoppt werden, um in den inaktiven Zustand zu gelangen. In diesem Zustand sind die verknüpften Datenmodule noch vorhanden, sodass das Modul ohne Änderungen wieder neu gestartet werden könnte. Beim Herunterfahren eines Moduls verhält sich ein Funktionsmodul anders als ein Datenmodul. Während das Datenmodul seine Ressourcen komplett frei gibt, bleibt das Funktionsmodul durch seine statische Architektur als Objekt vorhanden, verliert aber jegliche Verknüpfungen zu den Datenmodulen. In anderen Worten, das Funktionsmodul wird zurückgesetzt, anstatt es herunterzufahren. Der größte Unterschied zu einem Datenmodul ist ein weiterer Zustand für die Zeit, in der Analysen oder Berechnungen ausgeführt werden. Während dieser Zeit kann das Funktionsmodul nicht gestoppt werden. Nach der Berechnungsphase stehen Ergebnisse zur Verfügung und der Status des Moduls ist wieder auf aktiv gesetzt.

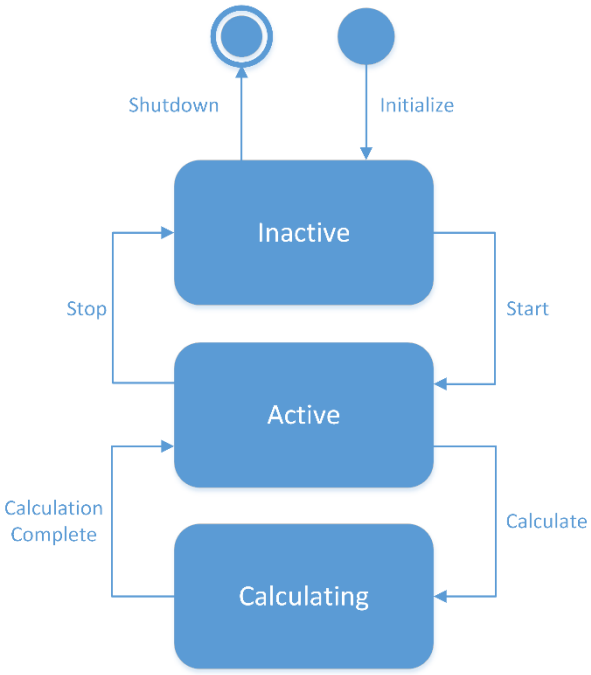

ABBILDUNG 24 - ZUSTANDSGRAPH EINES FUNKTIONSMODULS IM AR-FRAMEWORK

4.2.4 ZUGRIFFSMANAGER

Der Zugriffsmanager koordiniert den Zugriff auf Daten- und Funktionsmodule. Er bildet außerdem die Schnittstelle zum Benutzer des AR-Frameworks.

Bei der Beschreibung der ersten drei Komponenten wurden einige Restriktionen und Kontrollmechanismen erwähnt. Diese Managementfunktionalität befindet sich nicht innerhalb der Module, um die diese so einfach wie möglich zu halten. Stattdessen wird das Management im Zugriffsmanager gebündelt. Bis auf eine Ausnahme wird jede Aktion, die mit Hilfe des AR-Frameworks durchgeführt wird, vom Zugriffsmanager verwaltet. Über eine interne Zustandsliste kennt der Zugriffsmanager den Status sämtlicher Daten- und Funktionsmodule. Sowohl Daten- als auch Funktionsmodule registrieren sich beim Zugriffsmanager und melden Veränderungen an ihrem

Zustand. Möchte die Augmented Reality Anwendung zur Laufzeit auf ein Modul innerhalb des AR-Frameworks zugreifen, überprüft der Zugriffsmanager den Zustand des Moduls und kann gegebenenfalls den Zugriff verweigern. Neben der Verwaltung der Module, bildet der Zugriffsmanager gleichzeitig die Schnittstelle zum Nutzer des AR-Frameworks. Möchte ein Entwickler ein Datenmodul initialisieren oder ein Funktionsmodul nutzen, geht dies nur indem er mit dem Zugriffsmanager kommuniziert. Bevor der Zugriffsmanager genutzt werden kann, muss er initialisiert werden. Bei der Initialisierung des Zugriffsmanagers wird ein komplettes *AccessController*-Objekt erstellt. Dieses Objekt muss innerhalb des C#-Skripts verwaltet werden. Ein Zugriffsmanager kann mehrmals initialisiert werden, was dazu führt, dass mehrere Instanzen des AR-Frameworks angelegt werden. Für den normalen Anwendungsfall existiert jedoch nur ein Zugriffsmanager, der für alle Module verantwortlich ist. Module, die mit einem Zugriffsmanager erstellt wurden, sind von einem zweiten Zugriffsmanager aus nicht erreichbar. Der Vorteil zwei Zugriffsmanager zu nutzen, liegt in der parallelen Nutzung eines Funktionsmoduls mit unterschiedlichen Datenmodulen. Ist der Zugriffsmanager initialisiert, können folgende Operation mit ihm ausgeführt werden:

- Datenmodul anlegen
- Datenmodul löschen
- Datenmodul starten
- Verbindungsmodule einem Datenmodul zuweisen
- Funktionsmodul initialisieren
- Funktionsmodul nutzen

Die einzige Aktion innerhalb des AR-Frameworks, die nicht direkt über den Zugriffsmanager ausgeführt wird, ist das Instanziieren von Verbindungsmodulen. Diese können nach Belieben erstellt werden. Für die Nutzung eines Verbindungsmoduls muss diesem jedoch ein Datenmodul zugewiesen werden. Diese Aktion wird wiederum vom Zugriffsmanager verwaltet. Abbildung 25 zeigt, dass ein Entwickler für jede relevante Aktion im AR-Framework den Zugriffsmanager nutzen muss. Obwohl ein Verbindungsmodul direkt vom Unity Skript aus erstellt werden kann, ist der Zugriffsmanager nötig, um es mit den Daten- und Funktionsmodulen in Verbindung zu bringen.

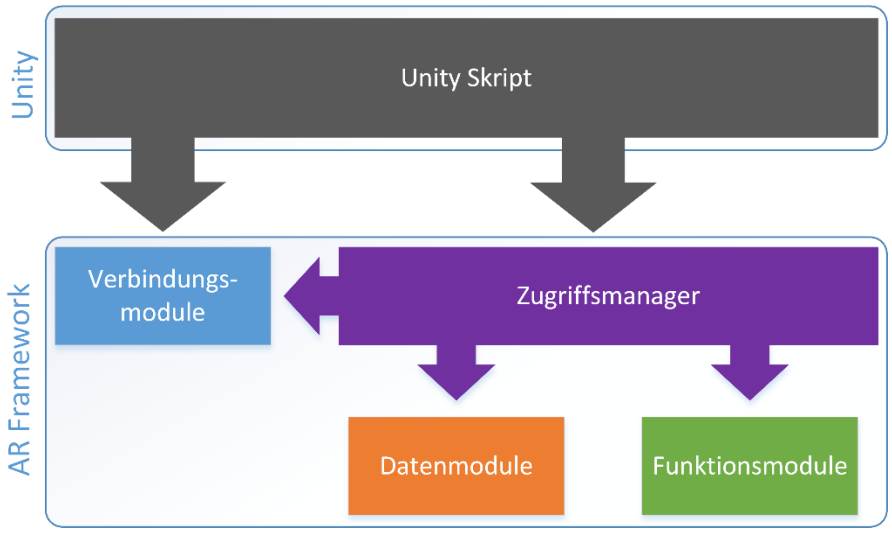

ABBILDUNG 25 - VERWALTUNG DES AR-FRAMEWORKS MIT HILFE DES ZUGRIFFSMANAGERS

5 SYSTEMMODELL

Dieses Kapitel vervollständigt die Übersicht über den Aufbau des AR-Frameworks. Nachdem in Kapitel 4 die Funktionsweise und der allgemeine Aufbau des Frameworks erklärt wurde, liegt in diesem Kapitel der Fokus auf der exakten Umsetzung dieser Modelle. Als Basis wird zu Beginn die Systemarchitektur beschrieben. Nachdem die einzelnen Komponenten des Systems beschrieben wurden, werden alle relevanten Klassen sortiert nach *Packages* im Detail erklärt.

5.1 SYSTEMARCHITEKTUR

Die Beschreibung der Systemarchitektur ist in Hardware- und Softwarearchitektur unterteilt. Die Hardwarearchitektur beschreibt nicht nur die Architektur zum Entwickeln einer Anwendung mit Hilfe des AR-Frameworks, sondern auch die potentiellen Geräte, auf denen eine erstellte Anwendung ausgeführt werden kann. Die Softwarearchitektur umfasst neben dem AR-Framework noch Komponenten aus Unity, die für das Framework notwendig sind.

5.1.1 HARDWAREARCHITEKTUR

Die Unity Umgebung bildet die Zielplattform des Frameworks. Das AR-Framework wird in Form einer DLL innerhalb von Unity ausgeführt. Für die Benutzung des Frameworks ist somit nur ein Computer nötig, der die Mindestanforderungen von Unity erfüllt. Diese sehen aktuell[6] folgendermaßen aus:

Betriebssystem:	Windows 7 SP1+, 8, 10, Mac OS X 10.8+
Grafikkarte:	DirectX 9 unterstützende Grafikkarte

Weitere Hardware Anforderungen hängen von der Größe des Projekts ab und sind von Unity nicht explizit genannt.

Da sowohl Unity Skripte, als auch das AR-Framework auf dem .NET Framework basieren, muss auf dem Entwicklungssystem zusätzlich die Version 3.5 des .NET Frameworks installiert sein. Je nachdem welche Datenmodule innerhalb des Frameworks benutzt werden, ergeben sich neue Anforderungen an die Hardware. Die bisher genannten Anforderungen beschränken sich auf die Entwicklungsumgebung für Augmented Reality Anwendungen mit dem AR-Framework. Eine erstellte Anwendung hat weitere Anforderungen. Der Anschluss einer Technologie zum Aufnehmen von Videomaterial ist zwingend notwendig für eine Augmented Reality Anwendung. Wird die Anwendung an einem mobilen Endgerät wie einem Tablet oder einem Smartphone ausgeführt, ist diese für gewöhnlich integriert. Für eine Ausführung an einem Computer ist die Installation einer Webcam nötig. Falls die in der Anwendung verwendeten Datenmodule eine Verbindung zu einer Werkzeugmaschine benötigen, muss das Zielgerät netzwerkfähig sein. Eine Zusammenfassung der notwendigen Hardware und der unterstützten Plattformen für Anwendungen ist Abbildung 26 zu entnehmen.

[6] Stand 20.09.2016

ABBILDUNG 26 - NOTWENDIGE UND UNTERSTÜTZTE HARDWARE KOMPONENTEN DES AR-FRAMEWORKS – TEILBILDQUELLE: DMGMORI[7], SAMSUNG[8]

5.1.2 SOFTWARE-ARCHITEKTUR

Die Software Architektur des AR-Frameworks lässt sich in zwei große Komponenten unterteilen, die jeweils mehrere kleine Komponenten besitzen. Abbildung 27 zeigt ein UML-Komponentendiagramm mit sämtlichen Komponenten des Systems. Nachfolgend werden diese Komponenten, aufgeteilt in zwei Gruppen, näher beschrieben und deren Funktion erläutert.

5.1.2.1 Unity-Komponente

Die *Unity*-Komponente enthält alle für das AR-Framework relevanten Komponenten innerhalb der Unity Umgebung. Die Komponente bietet eine *DLL-Import*-Schnittstelle an, die von der *AR-Framework*-Komponente genutzt wird. Über diesen Kommunikationsmechanismus wird das AR-Framework in Unity importiert und zur Verfügung gestellt. Dies ist die einzige Interaktion, die direkt über die beiden Hauptkomponenten ausgeführt wird. Alle weiteren Interaktionen geschehen innerhalb der Hauptkomponenten oder zwischen Komponenten aus den beiden Hauptkomponenten.

Die *MixedReality*-Komponente bietet Funktionalitäten für das Erzeugen einer Augmented Reality Anwendung. Der Großteil dieser Komponente besteht aus dem ARToolKit. Dieses stellt Skripte bereit, die von der *Scripting*-Komponente genutzt werden. Über diese Skripte können modellierte Elemente in Unity AR-fähig gemacht werden. Diese Elemente bezieht die *Scripting*-Komponente über die *3DModeling*-Komponente. Neben einer grafischen Möglichkeit 3D-Elemente, sogenannte Game Objekte zu erstellen, bietet diese Komponente eine Schnittstelle an, die innerhalb von Skripten benutzt wird, um zur Laufzeit neue Game Objekte zu erzeugen oder vorhandene Game Objekte zu manipulieren. Die *Scripting*-Komponente ist somit der zentrale Teil in Unity, der die anderen Komponenten verbindet. Darüber hinaus dient diese Komponente als Kommunikationskanal zum AR-Framework. Über die *DataModuleManagement*- und die *FunctionModuleUsage*-Schnittstelle greift die

[7] http://de.dmgmori.com/fachpresse/news/messevorbericht-emo-2013-hannover/166150
[8] http://www.samsung.com/de/news/product-/samsung-bringt-erste-fuer-breite-kaeuferschicht-entwickelte-vr-brille-in-den-handel

Scripting-Komponente direkt auf die *Controller*-Komponente innerhalb des AR-Frameworks zu. Über diese Schnittstellen kann Unity auf Daten und Informationen aus dem AR-Framework zugreifen.

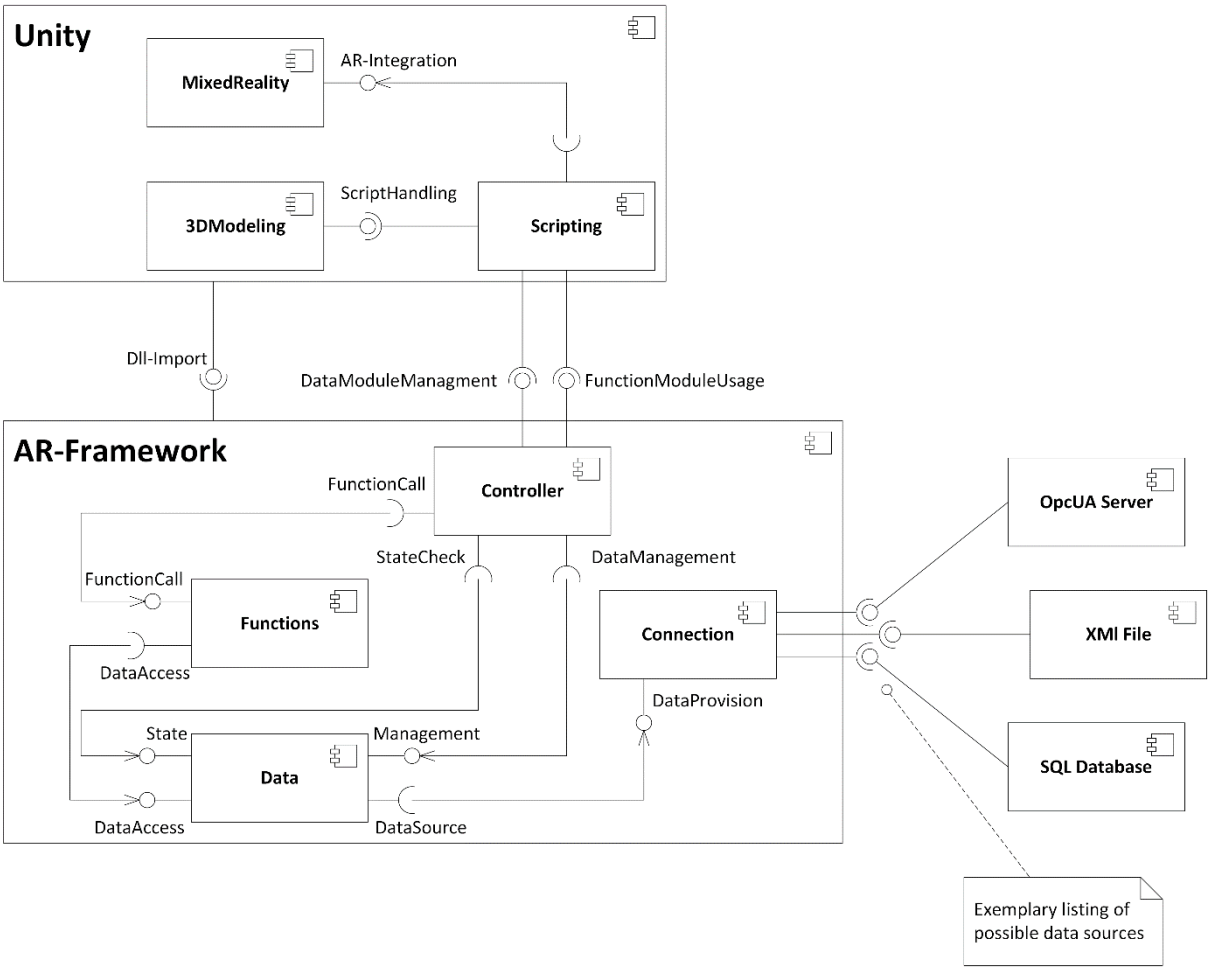

ABBILDUNG 27 – KOMPONENTENDIAGRAMM DES AR-FRAMEWORKS NACH UML 2.5

5.1.2.2 AR-Framework-Komponente

Die bereits angesprochene *Controller*-Komponente bildet die einzige Interaktionsmöglichkeit mit dem AR-Framework. Über sie können Datenmodule verwaltet und Funktionsmodule genutzt werden. Diese Schnittstelle verbirgt die Komplexität des Frameworks vor dem Nutzer und bietet gleichzeitig eine standardisierte Nutzung für alle Daten- und Funktionsmodule an. Innerhalb des AR-Frameworks verwaltet die *Controller*-Komponente den Zugriff auf Funktionsmodule in der *Functions*-Komponente sowie die Haltung von Daten innerhalb der *Data*-Komponente. Methoden der Datenmodule innerhalb der *Data*-Komponente werden über die *Management*-Schnittstelle nach außen angeboten und von der *Controller*-Komponente genutzt, um Datenmodule zu starten oder zu beenden. Zusätzlich bietet die *Data*-Komponente Zugang zum Status eines Datenmoduls. Dieser Status wird von der *Controller*-Komponente genutzt, um Abhängigkeiten eines Funktionsmoduls zu überprüfen. Die letzte Schnittstelle der *Data*-Komponente bietet Zugriff auf die Datenmodule innerhalb der Komponente. Diese Schnittstelle wird von der *Functions*-Komponente verwendet, um an benötigte Daten zu gelangen. Neben den drei Schnittstellen greift die *Data*-Komponente noch auf die Schnittstelle der *Connection*-Komponente zu. Diese Komponente bietet über die *DataProvision*-Schnittstelle verschiedene Verbindungsmodule an, um den *DataSource-Socket* zu füllen. Neben der *DataProvision*-Schnittstelle kann die *Connection*-Komponente mehrere Sockets für unterschiedliche Datenquellen

bieten. Am rechten Rand von Abbildung 27 sind drei beispielhafte Datenquellen abgebildet. Den letztendlichen Mehrwert des Frameworks bietet die *Functions*-Komponente. Über den *DataAccess-Socket* können Daten aus verschiedenen Datenmodulen innerhalb der Data Komponente bezogen werden. Die Funktionsmodule innerhalb der *Functions*-Komponente bieten die aus den Daten extrahierten Informationen über die *FunctionCall*-Schnittstelle an. Diese wird wiederum von der *Controller*-Komponente genutzt, um auf Wunsch des Entwicklers Informationen aus dem Framework an Unity zu leiten.

5.2 Allgemeiner Systementwurf

Um eine Stufe detaillierter zu zeigen, wie das AR-Framework aufgebaut ist, werden in diesem Kapitel die relevanten Klassen des AR-Frameworks, den Komponenten aus Abbildung 27 zugewiesen. Eine Komponente wird im UML-Diagramm aus Abbildung 28 durch ein *Package* ersetzt. Das *Common-Package* besitzt als einziges *Package* kein Gegenstück im Komponentendiagramm aus Abbildung 27. Dieses *Package* enthält ausschließlich Hilfsklassen für andere *Packages*. In diesem Kapitel wird nur der Sinn einer Klasse erklärt, nicht deren genauer Aufbau. Dieser Schritt erfolgt in Kapitel 5.3. Zusätzlich werden in diesem Kapitel beschreibenden Begriffe wie „Zugriffsmanager“ auf ihr Pendant im Quellcode des AR-Frameworks abgebildet.

Der Zugriffsmanager (*AccessController*) verwaltet die Datenmodule (*DataModule*) sowie die Funktionsmodule (*FunctionModule*) des Frameworks. In Abbildung 28 werden jeweils nur ein *DataModule* und ein *FunctionModule* innerhalb ihres Packages abgebildet. Die tatsächliche Anzahl der Module ist variabel. Beide Modultypen implementieren eine Schnittstelle, die es dem *AccessController* erlaubt, die Module zu steuern. Sobald ein Entwickler das Framework um ein *DataModule* oder ein *FunctionModule* erweitert, wie es in die entsprechende Komponente aufgenommen. Analog dazu verhalten sich Verbindungsmodule (*DataConnection*) aus dem *Connection-Package*. Auch hier können weitere Module hinzugefügt werden. Die *IDataConnection*-Schnittstelle wird implementiert, um einem *DataModule* Zugriff auf die standardisierten Verwaltungsmethoden einer *DataConnection* zu gewährleisten. *Enumerations* aus dem *Common-Package* werden von mehreren *Packages* genutzt. Der *eModuleState* beschreibt den derzeitigen Zustand eines *DataModules*, der *eDataConnectionState* den Zustand einer *DataConnection*. Der *AccessController* nutzt diese *Enumerations,* um die Zustände der *DataModules* zu überwachen. Der *eModuleType* wird genutzt, um dem *AccessController* zu übermitteln, welche Art *DataModule* initialisiert werden soll. Dieser interpretiert darauf die *Enumeration* und erstellt ein entsprechendes *DataModule*. Eine Liste von *eDataConnection*-Objekten innerhalb eines *DataModules* beschreibt die mit dem Modul kompatiblen *DataConnections*. Dieser grundlegende Überblick über den Aufbau des AR-Frameworks wird im folgenden Kapitel vertieft.

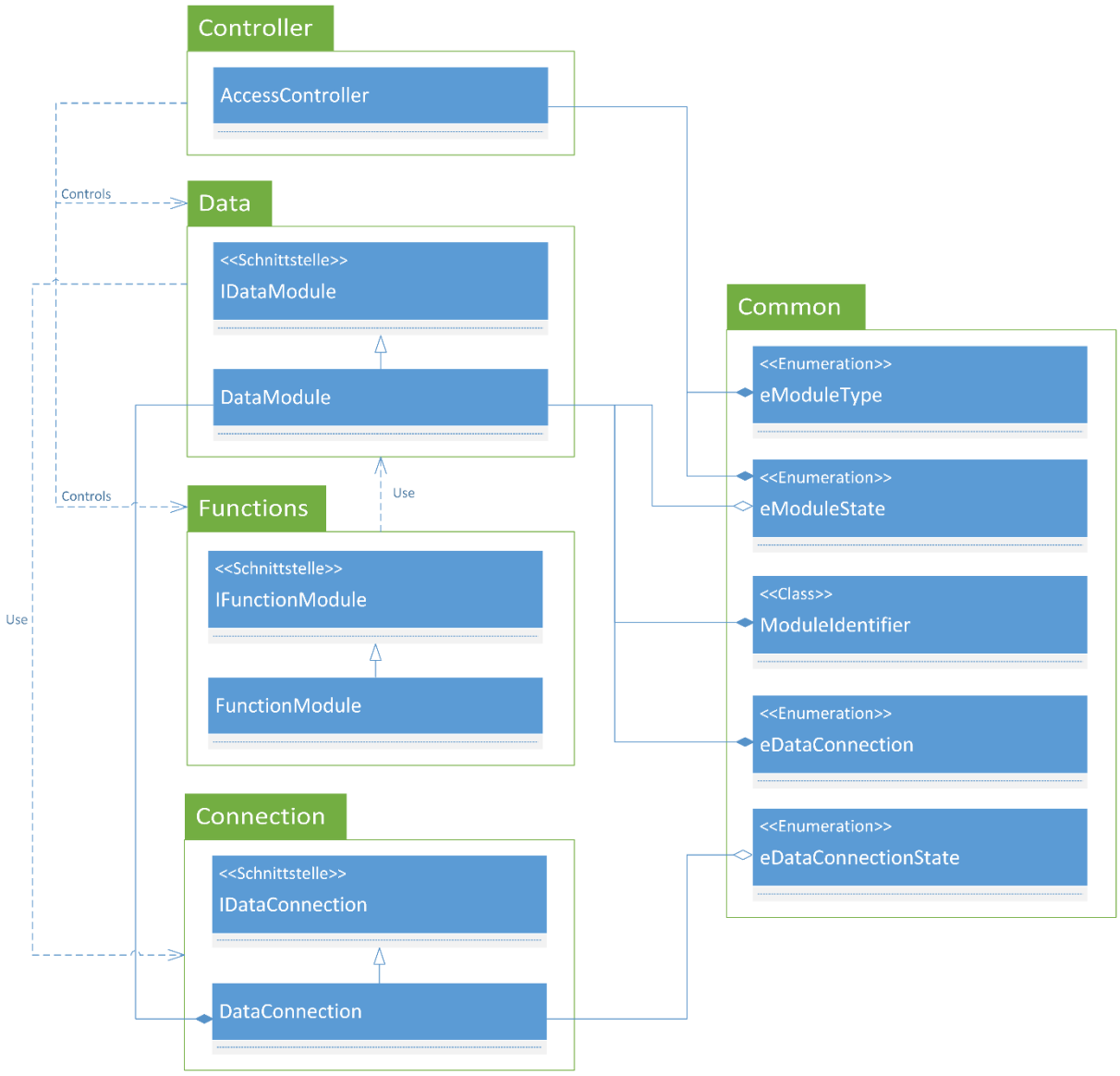

ABBILDUNG 28 - ÜBERSICHT DER KLASSEN INNERHALB VON PACKAGES IM AR-FRAMEWORK

5.3 KOMPONENTENSPEZIFISCHER ENTWURF

Dieses Kapitel ist der letzte Schritt in der immer detaillierter werdenden Erklärung des AR-Frameworks. Der Aufbau jeder, der in Abbildung 28 gezeigten Klassen wird in diesem Kapitel auf Basis eines oder mehrerer UML-Diagramme gezeigt und erklärt.

5.3.1 CONTROLLER

Abbildung 29 zeigt den konkreten Aufbau des *AccessControllers* in einem Klassendiagramm nach dem UML 2.5 Standard. Drei *private*-Datenstrukturen dienen dem *AccessController* zur Verwaltung der *DataModules* innerhalb des AR-Frameworks. *ModuleStates* ist ein *Dictionary*, das als Schlüssel *ModuleIdentifier* nutzt. Einzigartige *Integer* werden mit Hilfe des *IDCounter* erzeugt. Als zugehöriger Wert ist der Modulstatus für das Modul in Form einer *eModuleState-Enumeration* hinterlegt. Statusänderungen innerhalb eines Moduls führen zu einer Aktualisierung innerhalb des *Dictionary*. In einer *private-List* namens *DataModules* sind alle initialisierten *DataModule*-Objekte gespeichert. Über *public*-Methoden wie *StopDataModule* oder *StartDataModule* bietet der *AccessController* dem Frameworknutzer die Möglichkeit, aktiv *DataModules* zu verwalten. Direkten Zugriff auf die Module erhält der Frameworknutzer jedoch nicht. Alle *DataModules* liegen nur *private* vor und werden nicht

nach außen weitergegeben. Angelegte *DataConnections* können mit Hilfe der *SetDataConnection*-Methode einem *DataModule* zugeordnet werden. Da bei *DataConnections* nur lesender Zugriff möglich ist, kann eine *DataConnection* gleichzeitig an mehrere *DataModules* gebunden werden, ohne Konsistenzprobleme zu verursachen.

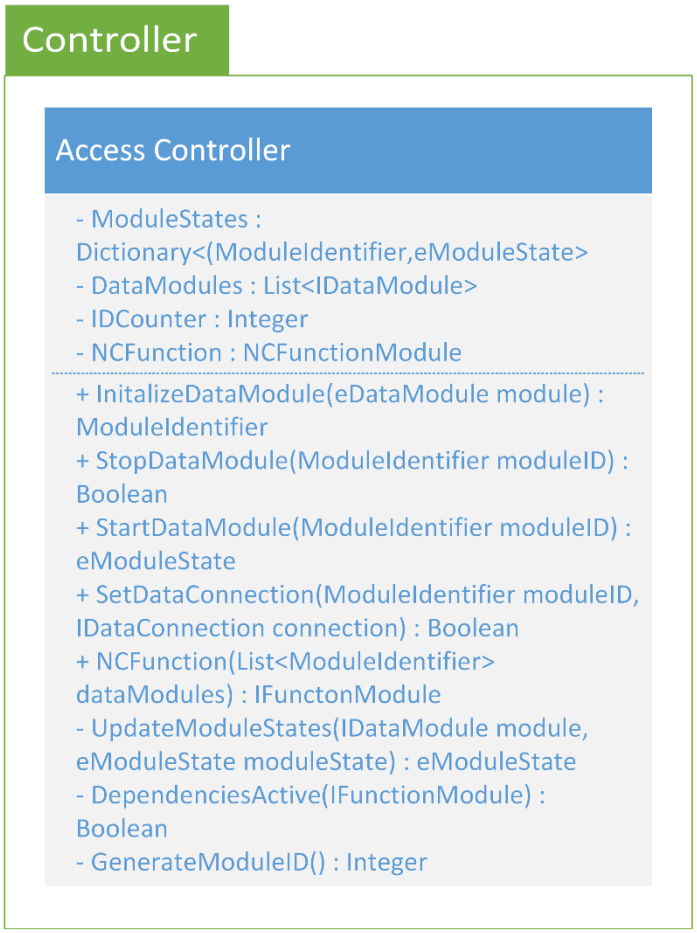

ABBILDUNG 29 – UML-KLASSENDIAGRAMM DES CONTROLLER-PACKAGE NACH UML 2.5

Jedes *FunctionModule* wird *private* und ohne Initialisierung im *AccessController* angelegt. Eine entsprechende *public*-Methode lässt den Frameworknutzer auf das *FunctionModule* zugreifen. Beim Aufruf dieser Methode werden *ModuleIdentifier* als Parameter übergeben. Der *AccessController* nutzt diese Objekte, um die *DataModules* in der *DataModules*-Liste zu finden und mit dem *FunctionModule* zu verknüpfen, sobald es initialisiert wurde. Eine Referenz auf das *FunctionModule* wird an den Frameworknutzer als Rückgabeparameter übergeben, sobald das *FunctionModule* korrekt initialisiert wurde. Um das *FunctionModule* zu initialisieren, prüft der *AccessController* über die *DependenciesActive*-Methode zuerst, ob sich alle angefragten *DataModules* in einem aktiven Zustand befinden. Ist dies der Fall, kann das *FunctionModule* initialisiert werden. Ist dies nicht der Fall, wird eine Fehlermeldung zurückgegeben.

Abbildung 30 zeigt eine beispielhafte korrekte Anwendung des *AccessControllers*. In diesem Beispiel wird als erster Schritt eine *DataConnection* zum Einlesen von NC-Daten erstellt. Als notwendiger Parameter wird der Pfad zu der Datei übergeben, die das NC-Programm enthält. Diese Aktion wird, wie bereits beschrieben nicht vom *AccessController* verwaltet. Im nächsten Schritt startet der Entwickler einen *AccessController* und legt ein *DataModule* an, indem er vom *AccessController* die *InitializeDataModule*-Methode aufruft. Als Parameter übergibt der Entwickler eine *Enumeration* vom

Typ *eDataModule*, damit der *AccessController* weiß, welches *DataModule* er starten soll. Nachdem dem *DataModule* die *DataConnection* zugewiesen wurde, kann das Modul gestartet werden. Dieser Modulstart führt automatisch zum Start der *DataConnection*. Innerhalb des *DataModules* wird jetzt ein Datenmodell aus den Daten der *DataConnection* erstellt. Um auf diese Daten zugreifen zu können, nutzt der Entwickler die *NCFunction*-Methode des *AccessControllers*. Diese Methode aktiviert das *FunctionModule*, das die Daten des *NCDataModules* zur Informationsgewinnung nutzt. Bevor das *FunctionModule* aktiviert werden kann, überprüft der *AccessController*, ob alle nötigen Abhängigkeiten des *FunctionModules* aktiv sind. Über die *DependenciesActive*-Methode greift der *AccessController* dafür auf seine interne Liste zu und überprüft ob das *DataModule* im aktiven Zustand ist. Da dies der Fall ist, wird das *FunctionModule* initialisiert und eine Referenz an den Entwickler zurückgegeben. Dieser kann jetzt das Modul nutzen um Informationen zu erhalten, die er in seiner Augmented Reality Anwendung benutzen kann.

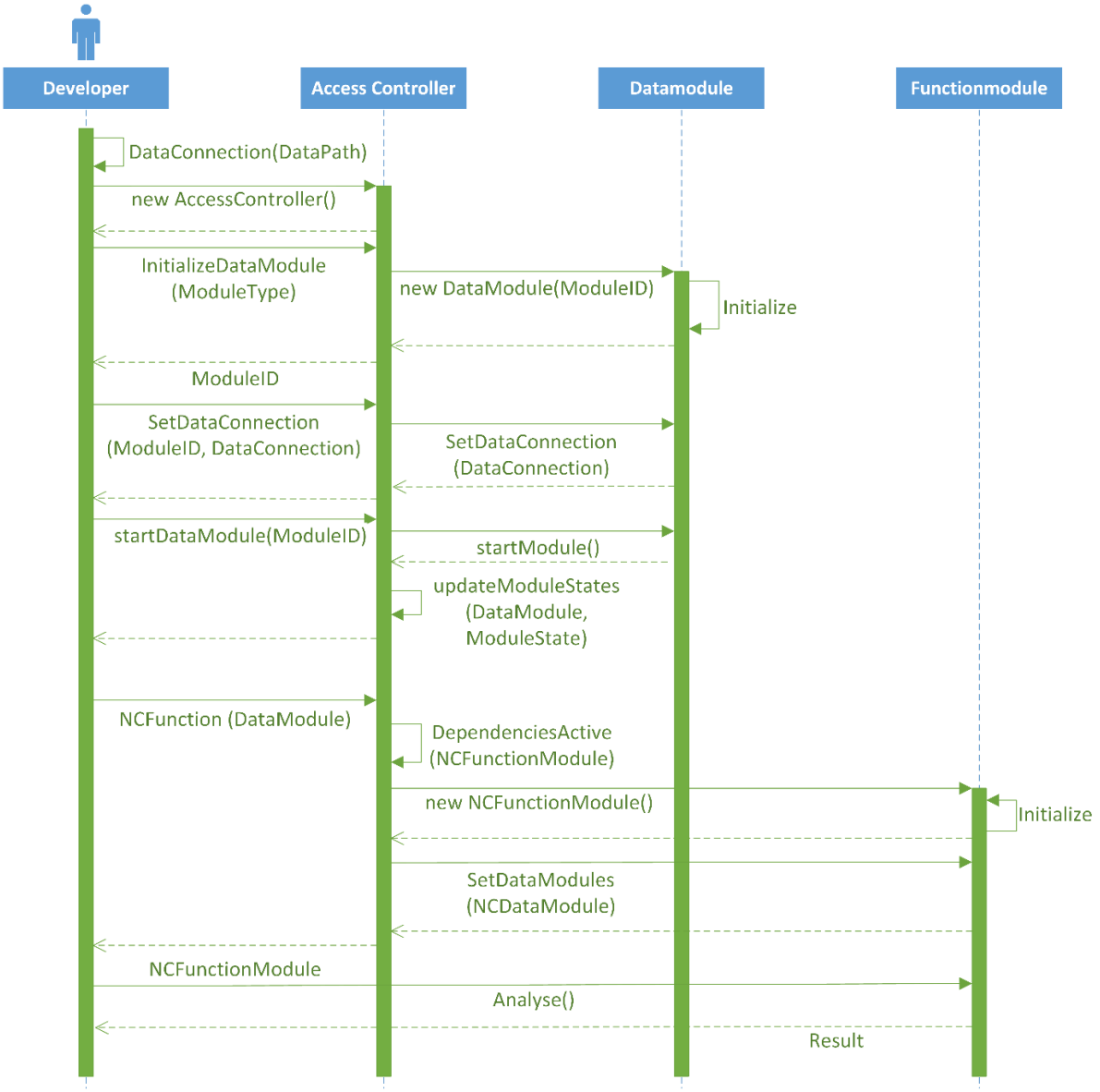

Abbildung 30 - Beispielhafter korrekter Ablauf einer Nutzung des AR-Frameworks

Abbildung 31 zeigt eine weitere mögliche Nutzung der oben angelegten Module, die zu einem Fehlerfall führt. Nachdem der Benutzer die Informationen über die NC-Daten erhalten hat, stoppt er

das *DataModule*. Der *AccessController* aktualisiert seine Zustandsliste indem er den Zustand des *DataModules* zuerst auf inaktiv setzt und dann die Referenz auf die Ressourcen frei gibt. Anschließend möchte der Entwickler ein weiteres Mal auf das *NCFunctionModule* zugreifen, um ein zweites Mal Informationen zu erhalten. Das *FunctionModule* versucht vergeblich das *DataModule* zu erreichen und gibt eine Fehlermeldung an den Entwickler zurück. Wäre nicht das *DataModule*, sondern nur die *DataConnections* innerhalb des *DataModules* gestoppt worden, hätte das *FunctionModule* weiterhin auf Daten zugreifen können. Das *DataModule* hätte dem *FunctionModule* signalisiert, dass die Daten eventuell veraltet sind.

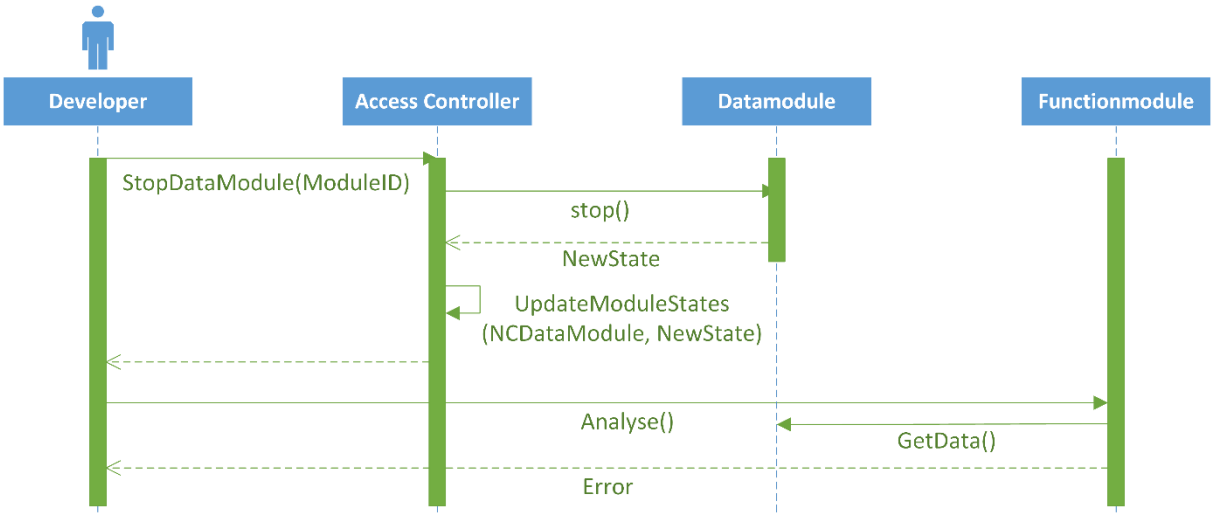

ABBILDUNG 31 - BEISPIELHAFTER FEHLERHAFTER ABLAUF EINER NUTZUNG DES AR-FRAMEWORKS

Beide Ablaufdiagramme aus Abbildung 30 und 31 verdeutlichen, dass der *AccessController* die Schnittstelle zum Entwickler für die Verwaltung von Modulen innerhalb des AR-Frameworks ist. *FunctionModules* können unabhängig vom *AccessController* genutzt werden. Die nötige Initialisierung der *FunctionModules* ist jedoch ausschließlich über den *AccessController* möglich.

5.3.2 DATA

Das *Data*-Package enthält neben sämtlichen *DataModules* eine Schnittstelle, die jedes *DataModule* implementieren muss. Die *IDataModule*-Schnittstelle aus Abbildung 32 beinhaltet Methoden, die vom *AccessController* genutzt werden, um das *DataModule* zu verwalten. Die Methoden *StartModule* und *StopModule* werden genutzt, um den Lebenszyklus eines *DataModules* zu kontrollieren. Beide Funktionen geben einen *booleschen* Wert zurück, um zu signalisieren, ob die angeforderte Aktion erfolgreich ausgeführt wurde. Über *GetModuleState* kann der *AccessController* jederzeit den aktuellen Status des Moduls abfragen. *GetModuleType* gibt eine *Enumeration* zurück, die den Typ des Moduls enthält. Eine eindeutige Identifizierung des *DataModules* ist über den *ModuleIdentifier* möglich, den der *AccessController* über *GetModuleID* abfragen kann. Neben diesen Methoden zum Abfragen von Informationen über ein *DataModule* gibt es zwei weitere Methoden, um das *DataModule* aktiv zu beeinflussen. *SetModuleState* gibt dem *AccessController* die Möglichkeit, den Status eines *DataModules* zu ändern. Diese Methode wird benötigt, da sich der Status eines *DataModules* nicht nur beim Starten und Stoppen ändern kann, sondern zum Beispiel auch, wenn eine verknüpfte *DataConnection* beendet wird. Als Rückgabewert für diese Methode wird nicht der Erfolg des Aufrufs verwendet, sondern der aktuelle Status des Moduls. Realisiert der *AccessController* eine solche

Änderung, kann er den Status des *DataModules* auf inaktiv setzen, ohne es komplett zu beenden. Die zweite Möglichkeit ein *DataModule* zu beeinflussen, ist die Zuweisung einer *DataConnection*. *SetDataConnection* ermöglicht diesen Vorgang. Wird eine *DataConnection* an ein *DataModule* übergeben, muss das *DataModule* entscheiden, ob es diesen *DataConnection*-Typ unterstützt oder nicht. Ein *boolescher* Rückgabewert signalisiert dem *AccessController*, ob eine *DataConnection* akzeptiert wurde oder nicht. Drei vorgegebene Events können vom *AccessController* genutzt werden, um den Vorgängen innerhalb eines *DataModules* zu folgen. Das *StateChanged-Event* wird ausgelöst, sobald sich der Status innerhalb eines *DataModules* ändert. Das Starten und Stoppen eines Moduls löst die *Events OnModuleStart* und *OnModuleStop* aus.

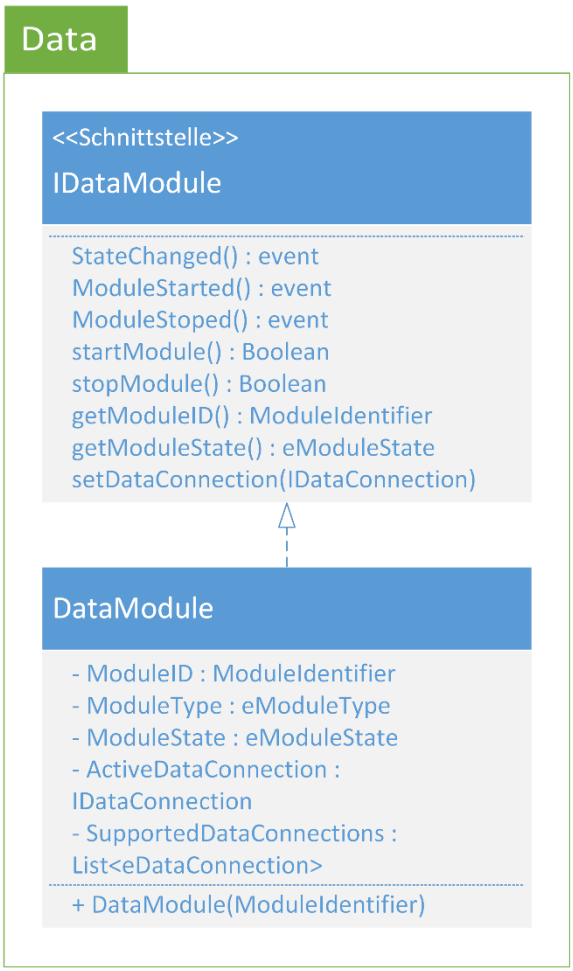

ABBILDUNG 32 – UML-KLASSENDIAGRAMM DES DATA-PACKAGE NACH UML 2.5

Neben den Kontrollmethoden aus dieser Schnittstelle hat ein *DataModule* weitere Variablen und Methoden, die für eine konsistente Struktur innerhalb der *DataModules* sorgen. Damit ein Entwickler sich an dieser Struktur orientieren kann, bietet ein Visual Studio Template eine Vorlage für jedes neue *DataModule*. Die Struktur, die diese Vorlage bietet, wird ebenfalls in Abbildung 32 in der *DataModule* Klasse gezeigt. Jedes *DataModule* hat nach dieser Vorlage zwei *private*-Variablen zum Speichern des eigenen Status und des Identifizierungsobjekts. Das *ActiveDataConnection*-Objekt eines *DataModules* beinhaltet die derzeit gesetzte *DataConnection*. *SupportedDataConnections* ist eine Liste, in der statisch alle unterstützten *DataConnection*-Typen in Form von *Enumerations* aufgezählt werden. Diese Liste wird initial befüllt und nicht mehr verändert. Der Modulentwickler legt in dieser Liste fest, welche *DataConnection*-Typen in diesem *DataModule* genutzt werden können. Neben den vordefinierten

Objekten, gibt die Vorlage auch die Struktur des Konstruktors vor. Jeder Konstruktor sollte als einzigen Übergabeparameter einen *ModuleIdentifier* haben. Diese werden vom *AccessController* erstellt und bei der Initialisierung eines *DataModules* übergeben.

Neben den Methoden aus der Schnittstelle und den Variablen aus der Vorlage kann ein Entwickler sein *DataModule* frei gestalten. Um alle *DataModules* konsistent zu halten, sollten jedoch zumindest die syntaktischen Vorgaben aus der Vorlage übernommen werden.

5.3.3 *Functions*

Die *Functions*-Komponente ist analog zur *Data*-Komponente aufgebaut. Wie in Abbildung 33 sichtbar, beinhaltet die Komponente die *IFunctionModule*-Schnittstelle sowie die implementierten *FunctionModules*. Wie beim *DataModule* wird in Abbildung 33 ein beispielhaftes *FunctionModule* ohne jegliche Methoden mit zusätzlicher Funktionalität angezeigt. Das *FunctionModule* enthält lediglich die grundlegenden Methoden aus der Schnittstellenimplementierung und dem Visual Studio Template.

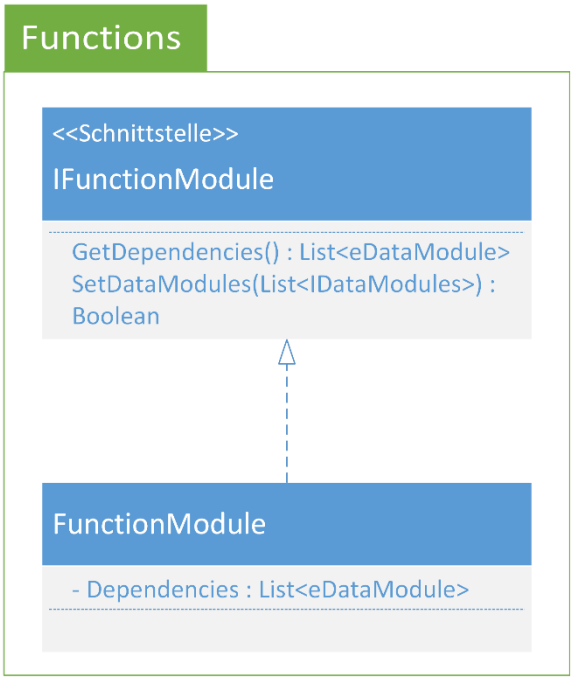

Abbildung 33 – UML-Klassendiagramm des Functions-Package nach UML 2.5

Die Schnittstelle für *FunctionModules* ist bewusst klein gehalten, da ein *FunctionModule* wenig Verwaltungsaufwand erfordert. Über die *GetDependencies*-Methode können alle zur Nutzung dieses Moduls benötigten *DataModules* abgefragt werden. Diese *DataModules* werden in Form einer Liste von *Enumerations* zurückgegeben. Über *SetDataModules* können einem *FunctionModule* die benötigten DataModules zugewiesen werden. Da jedes *FunctionModule* eine beliebige Anzahl an Abhängigkeiten haben kann, wird eine Liste mit undefinierter Größe übergeben. Der Entwickler eines *FunctionModules* hat die Aufgabe, die *DataModules* aus dieser Liste entsprechend zu speichern. Mehr als diese zwei Methoden werden nicht benötigt, um *FunctionModules* ansteuern zu können. Das Starten und Stoppen eines *FunctionModules* erfolgt bei der Initialisierung beziehungsweise beim Löschen der Referenz auf das Objekt, da es pro *AccessController* Instanz jedes *FunctionModule* nur einmal gibt.

Die Visual Studio Vorlage für ein *FunctionModule* fällt ebenfalls sehr klein aus. Eine private Liste mit allen Abhängigkeiten in Form von *eModuleType-Enumerations* speichert alle *DataModule* -Typen, die ein *FunctionModule* benötigt. Diese Liste wird während der Entwicklung des *DataModules* angelegt und bleibt zur Laufzeit unverändert. Neben dieser Variable bietet die Vorlage eine auskommentierte Anleitung zur Verwaltung von *DataModules* innerhalb des *FunctionModules*. Jedes *DataModule* aus der Dependencies-Liste sollte ohne Initialisierung im *FunctionModule* vorliegen. Werden *DataModules* über die *SetDataModules*-Methode der I*FunctionModule*-Schnittstelle übergeben, können diese den vorhandenen *DataModules* zugeordnet werden. Abbildung 34 zeigt eine beispielhafte Umsetzung für dieses Vorgehen.

```
public bool SetDataModules(List<Data.IDataModule> dataModules)
{
    foreach (Data.IDataModule module in dataModules)
    {
        switch (module.getModuleType())
        {
            case Common.eModuleType.NCModule:
                _dataModule = (Data.NCModule)module;
                break;
            default:
                return false;
        }
    }
    return true;
}
```

ABBILDUNG 34 - BEISPIEL FÜR DIE UMSETZUNG DER SETDATAMODULES-METHODE DER IFUNCTIONMODULE-SCHNITTSTELLE

5.3.4 CONNECTION

Der grundlegende Aufbau der *Connection*-Komponente, sichtbar in Abbildung 35, deckt sich mit dem Aufbau der *Data*- und *Functions*-Komponente. Eine zu implementierende Schnittstelle versorgt die *DataConnections* mit ausreichender Funktionalität, um standardisiert von jedem *DataModule* genutzt werden zu können. Die einzigen Methoden der Schnittstelle sind die *Start*- und die *Stop*-Methode. Die *Start*-Methode startet in der *DataConnection* die Kommunikation mit der Datenquelle. Es ist der Auftrag des Programmierers einer *DataConnection*, die Nutzung der unterliegenden Technologie so zu vereinfachen, dass die *Start*-Methode keine Parameter benötigt, um Daten zu beziehen. Die *Stop*-Methode hat je nach Art der *DataConnection* eine unterschiedliche Semantik. Bei einem kontinuierlichen Datenempfang wird der Kommunikationskanal zwischen Datenquelle und *DataConnection* geschlossen. Der Zustand der bereits empfangenen Daten bleibt erhalten, ändert sich jedoch nicht. Der Status der *DataConnection* wird anschließend auf inaktiv gesetzt. Bei einer *DataConnection* mit einmaligem Datenempfang wird ausschließlich der Zustand der *DataConnection* auf inaktiv gesetzt. Bei beiden Empfangsarten ist zu beachten, dass die bereits empfangenen Daten weiterhin in der *DataConnection* gespeichert sind. Sollte ein Fehler in der *DataConnection* dafür sorgen, dass vom aktiven in den inaktiven Zustand gewechselt wird, so kann ein *DataModule* trotzdem weiterhin Daten beziehen. Es liegt am Entwickler der *DataConnection*, eine Methodik zu implementieren, die auftretende Fehler repariert und die *DataConnection* wieder in den aktiven Zustand befördert.

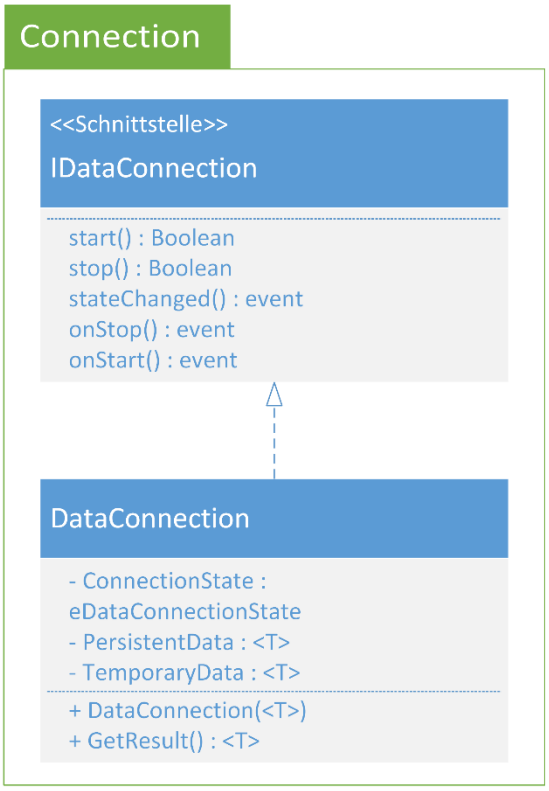

Abbildung 35 – UML-Klassendiagramm des Connection-Package nach UML 2.5

Während der Reparaturzeit sind jedoch potentiell alte Datensätze abrufbar. Diese Heruntersetzung der Servicequalität macht *DataConnections* robust gegenüber Fehlern. Nicht jedes *DataModule* möchte potentiell veraltete Daten verwenden. Aus diesem Grund sind Entwickler von *DataConnections* angewiesen, bei der Ergebnisabfrage zu signalisieren, ob die Werte aktuell sind oder ob sich die *DataConnection* momentan in einem inaktiven Zustand befindet. Um zu ermöglichen, dass zu jederzeit konsistente Daten abgerufen werden können, sind zwei Datensätze nötig. Ein vollständiger, eventuell veralteter Datensatz bietet zuverlässigen Zugriff auf konsistente Daten. Ein temporärer Datensatz wird benutzt, um den aktuellsten Datensatz zwischenzuspeichern. War der Prozess der Datenbeschaffung erfolgreich, wird der konstante Datensatz mit dem temporären Datensatz überschrieben. Gab es einen Fehler während der Datenbeschaffung ist der persistente Datensatz weiterhin konsistent, jedoch potentiell veraltet. Als Rückgabewert bei der Datenabfrage wird ausschließlich der persistente Datensatz verwendet. Der temporäre Datensatz wird gelöscht, sobald der Datenbeschaffungsprozess abgeschlossen ist. Hierbei spielt es keine Rolle, ob der Prozess erfolgreich oder fehlerhaft war.

Analog zum *IDataModule* besitzt auch die *IDataConnection* Schnittstelle drei Events auf die ein *DataModule* reagieren kann. Über *OnStop* und *OnStart* kann das *DataModule* auf unplanmäßige Prozesse reagieren. Ein unplanmäßiger Stopp könnte durch einen Fehler beim Lesen von Daten auftreten. Ein unplanmäßiger Start kann durch eine Selbstreparatur der *DataConnection* nach einem Fehlerfall hervorgerufen werden. Das *StateChanged Event* kann von einem *DataModule* genutzt werden, um zu prüfen in welchem Zustand sich die *DataConnection* zum Zeitpunkt des Zugriffs auf die Daten befindet. Befindet sich die *DataConnection* durchgängig im aktiven Zustand, kann das *DataModule* davon ausgehen, dass erhaltene Daten aktuell sind. Ist die *DataConnection* in einem inaktiven Zustand, könnten erhaltene Daten veraltet sein.

Neben der Schnittstelle gibt die Visual Studio Vorlage Vorschläge für den Aufbau einer *DataConnection*. In einer *private*-Variable wird der Status der *DataConnection* in Form einer *eDataConnectionState Enumeration* festgehalten. Dieser interne Status kann über die *IDataConnection*-Schnittstelle an das zugewiesene *DataModule* übergeben werden. Der *DataConnectionType* speichert den Typ der jeweiligen *DataConnection* in Form einer *eDataConnection Enumeration*. Dieser Typ kann von einem *DataModule* über die *IDataConnection*-Schnittstelle abgefragt werden, sobald die *DataConnection* dem *DataModule* zugewiesen wird. Die *DataConnection*-Vorlage gibt neben diesen konkreten Variablen zusätzliche Richtlinien zum Aufbau des Konstruktors vor. Sämtliche Informationen die eine *DataConnection* braucht um die unterliegende Technologie zu nutzen, sollten als Parameter übergeben werden. Da jede Technologie unterschiedliche Parameter benötigt, gibt es hier keine feste Vorschrift. Für die Ausgabe der erhaltenen Daten gibt es aus dem gleichen Grund keine strengen Vorgaben. Jede Technologie liefert ihre Daten in einem anderen Format. Nach den Richtlinien der Visual Studio Vorlage ist ein *DataConnection*-Entwickler jedoch dazu angehalten, Daten über die *GetResult*-Methode an die *DataModules* weiter zu geben. Über den Aufbau des Rückgabeparameters gibt es jedoch keine Einschränkungen. Werden alle Richtlinien eingehalten, besteht die Schnittstelle zu jeder *DataConnection* aus drei essentiellen Methoden. Starten, Stoppen und das Abfragen der Daten kann mit insgesamt lediglich drei Aufrufen erledigt werden.

5.3.5 COMMON

Diese Komponente besitzt ausschließlich Hilfsklassen, die für eine bessere Übersicht und für höhere Codequalität sorgen. In Abbildung 36 sind alle Klassen im Common Paket in einem UML 2.5 Klassendiagramm zu sehen. Verwendete *eModuleType* und *eDataConnection* Inhalte entsprechen einer prototypischen Realisierung einer Augmented Reality Anwendung mit Hilfe des AR-Frameworks. Genauere Informationen hierzu befinden sich in Kapitel 6. Die zwei *Enumerations eModuleState* und *eDataConnectionState* repräsentieren den Zustand der *DataModules*, beziehungsweise der *DataConnections*. *Enumerations* anstatt feste *Strings* zu nutzen, ist ein Mittel, um Fehler innerhalb der Anwendung durch Schreibfehler zu vermeiden. Nach Cwalina und Abrams [11] gibt es nur wenige Fälle, in denen feste *Strings* an Stelle von *Enumerations* genutzt werden sollten. Die zwei *Enumerations eDataType* und *eDataConnection* repräsentieren den Typ von *DataModules* und *DataConnections*. Bei der Nutzung einer von beiden Komponenten muss häufig überprüft werden, ob diese Komponente mit der anderen kompatibel ist. *Enumerations* bilden die angezeigten Namen intern auf Integer ab und sind deshalb eine ressourcensparende Variante, um einen Datentyp zu repräsentieren. Beide *Enumerations* müssen nach dem Erstellen eines *DataModules* oder einer *DataConnection* erweitert werden. Der damit zusammenhängende Wartungsaufwand ist der Nachteil dieser Methode. Die letzte Hilfsklasse in dieser Komponente ist der *ModuleIdentifier*. Diese Klasse ist ein Konstrukt aus einem *eModuleTyp* und einem *Integer*. Diese Kombination dient als einzigartiger Identifikationsmechanismus für *DataModules*.

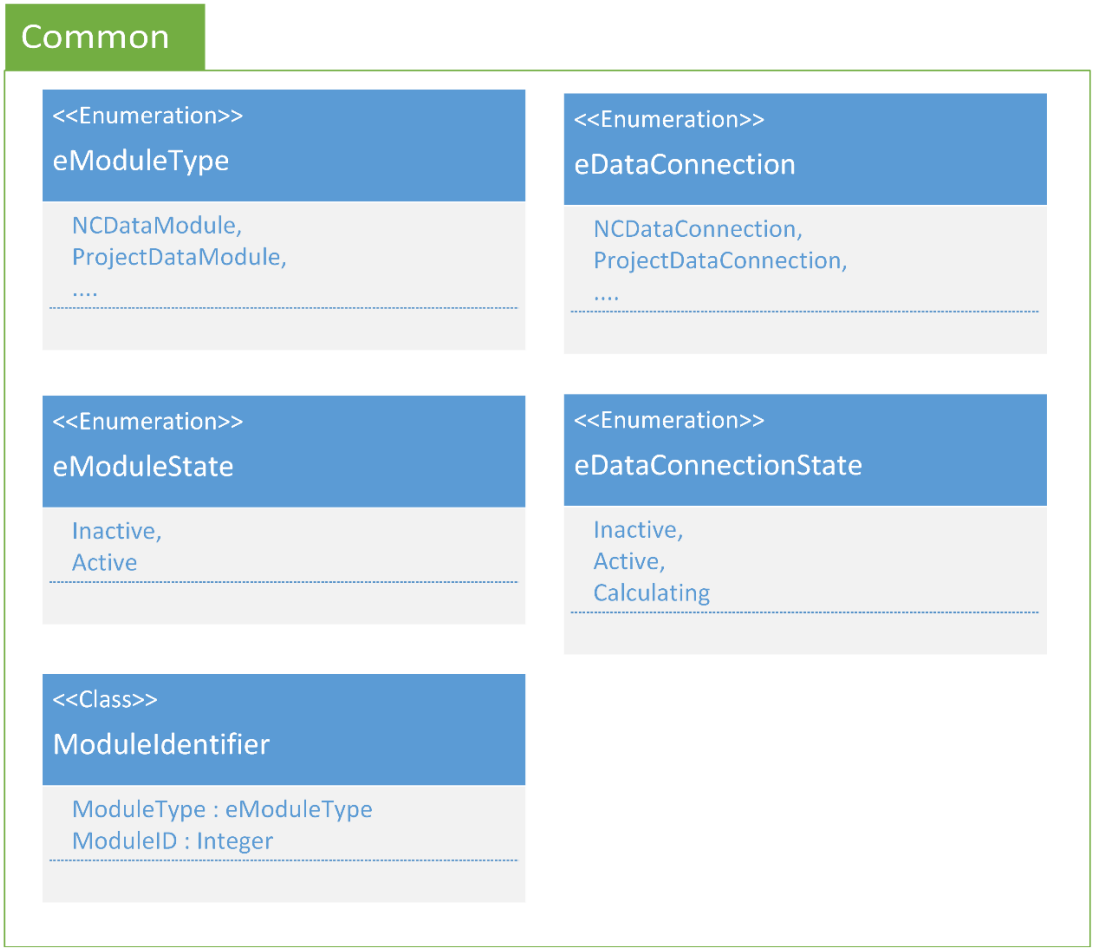

ABBILDUNG 36 – UML-KLASSENDIAGRAMM DES COMMON-PACKAGE NACH UML 2.5

6 REALISIERUNG

Um zu validieren, ob das Framework die gestellten Anforderungen erfüllt, wird in Zusammenarbeit mit einem Industriepartner eine Augmented Reality Anwendung mit Hilfe des AR-Frameworks erstellt. Nachfolgend wird erläutert, welche Funktionalität die Augmented Reality Anwendung hat und wie diese in der Produktion genutzt werden kann. Anschließend wird der Aufbau aller erstellten Datenmodule, Funktionsmodule und Verbindungsmodule innerhalb des Frameworks gezeigt. Hierfür werden Klassendiagramme nach UML-Notation verwendet. Nachdem die Konstruktion der einzelnen Module geklärt ist, dokumentieren Bilder die verschiedenen Zustände der Anwendung während der Entwicklung.

6.1 GRUNDLAGEN ZUR ANWENDUNG

Der Industriepartner besitzt Fräsmaschinen, die aus Blechplatten zweidimensionale Formen schneiden. Auf einer Platte können sich unterschiedliche Formen befinden. Diese Formen sind oft nicht gruppiert, da eine optimale Anordnung der Formen den Ausschuss der Blechplatten minimieren kann. Die unterschiedlichen Formen können verschiedenen Projekten angehören. Je nach Projektzugehörigkeit muss ein Mitarbeiter die fertigen Teile auf verschiedene Transportmittel legen. Ein Transportmittel kann eine Europalette, eine Kiste oder Ähnliches sein. Zum jetzigen Zeitpunkt besitzt der Mitarbeiter ein Dokument, mit dessen Hilfe er die nötigen Informationen zur Bearbeitung der Blechteile erhält. In diesem Dokument sind die eindeutige, textuelle Bezeichnung des Blechteils und das dazugehörige Transportmittel verzeichnet. Der Mitarbeiter versucht entsprechend den Vorgaben, viele Blechteile in möglichst geringer Zeit auf das korrekte Transportmittel zu legen. Dieser Prozess ist weder effizient noch sicher. Eine Augmented Reality Anwendung soll versuchen, diese beiden Aspekte zu verbessern.

Die Effizienz soll durch die Verwendung von grafischen Darstellungen verbessert werden. Wenn der Mitarbeiter nicht mehr nur die Bezeichnung des Blechteils und dessen zugehöriges Transportmittel kennt, sondern eine grafische Verbindung zwischen beiden Elementen sieht, kann er mit einem Blick mehr Blechteile verarbeiten. Diese These soll mit einer Augmented Reality Anwendung überprüft werden. Dafür werden alle ausgeschnittenen Blechteile auf dem Blech mit visuellen Effekten erweitert. Ein Rahmen in einer Farbe, die einem bestimmten Transportmittel zugewiesen ist, soll zeigen, welches Teil auf welches Transportmittel gelegt werden muss. Diese Funktionalität wird in einer Anwendung umgesetzt, die mit Hilfe des AR-Frameworks entwickelt wird.

Um die Sicherheit des Prozesses weiter zu erhöhen, kann ein Tracking-Verfahren genutzt werden, um zu überprüfen, welche Blechteile der Mitarbeiter in die Hand nimmt. Eine beispielhafte Anwendung könnte dem Mitarbeiter vorgeben, dass er als nächstes alle Teile einer bestimmten Farbe zum zugehörigen Transportmittel legen soll. Anschließend kann erfasst werden, welche Teile der Mitarbeiter in die Hand nimmt. Sollte er ein falsches Teil in die Hand nehmen, kann ihm dies durch akustisches oder visuelles Feedback aufgezeigt werden. Eine erhöhte Sicherheit des Prozesses gehört jedoch nicht zu den Anforderungen an die Anwendung, die mit Hilfe des AR-Frameworks erstellt wird. Sollte sich zeigen, dass die Effizienz durch die Anwendung erhöht werden kann, ist es möglich, die Anwendung um diesen Teil zu erweitern.

Für die virtuelle Einfärbung von ausgeschnittenen Blechteilen sind mehrere Informationen erforderlich. Zum einen muss bekannt sein, welche Formen aus dem Blech gefräst werden. Insbesondere die Kontur der Formen ist relevant, um diese virtuell nachzubilden. Diese virtuellen Elemente müssen anschließend am korrekten Ort in der realen Welt abgebildet werden. Es ist

entsprechend nötig zu wissen, wo sich die ausgefrästen Teile auf dem Blech befinden. Sowohl die Form der Blechstücke, als auch die Position auf dem Werkstück kann dem NC Code entnommen werden, der die Fräsmaschine steuert. Um die Konturen in der richtigen Farbe darzustellen, ist es notwendig zu wissen, welche Blechteile zu welchem Projekt gehören. Zusätzlich muss bekannt sein, welches Transportmittel für welches Projekt benutzt wird. Diese Daten liegen in Form des Dokuments vor, das der Mitarbeiter nutzt, um seine Informationen zu beziehen.

6.2 UMSETZUNG

Aus den Grundlagen zur Anwendung geht hervor, dass für die Umsetzung der Augmented Reality Anwendung zwei verschiedene Datenquellen genutzt werden müssen. Der NC Code für Darstellung und Ausrichtung der virtuellen Konturen sowie das Projektdokument für die Zuordnung der Blechteile zu einem Projekt und dessen Transportmittel. Für beide Datenquellen wird nach den Richtlinien des AR-Frameworks jeweils ein Verbindungsmodul angelegt. Um die Daten aus den Verbindungsmodulen zu verarbeiten, wird jeweils ein Datenmodul implementiert. Ein Funktionsmodul nutzt beide Datenmodule, um die Daten zu verknüpfen, nötige Informationen zu generieren und diese für die Nutzung in Unity vorzubereiten. In den folgenden Abschnitten werden alle Module im Detail erklärt.

6.2.1 VERBINDUNGSMODULE

Die *NCDataConnection* greift mit Hilfe einer Bibliothek auf das NC-Programm zu und gibt die erhaltenen Daten an das *NCDataModule* weiter. Die *ProjectDataConnection* liest Projektdaten ein. Diese Daten müssen ein bestimmtes Format besitzen, um maschinell verwertbar zu sein. Der genaue Aufbau der Module wird nachfolgend durch UML-Klassendiagramme beschrieben.

6.2.1.1 NCDataConnection

Diese DataConnection nutzt einen *NCParser*, der vom Industriepartner zur Verfügung gestellt wird. Dieser *Parser* kann NC-Code einlesen und dessen Inhalt analysieren. Es ist möglich, den Quellcode des NC-Programms auszulesen, jedoch ist die Analyse des Codes für diesen Anwendungsfall interessanter. Der *Parser* ermöglicht es, eine Punktfolge zu generieren, welche die Formen beschreibt, die von der Maschine mit Hilfe des NC-Codes gefräst werden. Zusätzlich können die Position und die Rotation der einzelnen Formen auf dem Blech bestimmt werden. Die *NCDataConnection* greift diese Informationen aus dem *NCParser* ab und bietet sie einem *DataModule* an.

Der Aufbau der *NCDataConnection* ist größten Teils durch das *AR-Framework* vorgegeben. Alle Methoden der *IDataConnection*-Schnittstelle sind implementiert. Um das Klassendiagramm in Abbildung 37 nicht zu unübersichtlich erscheinen zu lassen, wurden diese Methoden aus der Schnittstelle entfernt. Neben den Schnittstellenmethoden sind Methoden aus der DataConnection Vorlage implementiert. Der *ConnectionState* sowie der *DataConnectionType* sind Teil jeder *DataConnection*. Der *FilePath* speichert die Adresse der zu analysierenden NC-Code Datei. Dieser Pfad wird dem Konstruktor übergeben. Die Vorlage beschreibt ebenfalls die persistenten und temporären Datensätze innerhalb einer *DataConnection*. *PersistentNCData* und *TemporaryNCData* sorgen wie in Kapitel 5.3.4 beschrieben für einen robusten Zugriff auf die *DataConnection*. Neben dem Konstruktor besitzt die *NCDataConnection* Klasse eine weitere *public*-Methode. *GetResult* kann vom angehängten *DataModule* genutzt werden, um auf *PersistentNCData* zuzugreifen. Beim Starten der *NCDataConnection* wird die *LoadNCFile*-Methode aufgerufen. Diese startet den unterliegenden *NCParser* und lässt ihn die NC-Daten aus der Datei auslesen, die unter dem vorgegebenen Pfad zu finden ist. Kann der *NCParser* diese Datei ohne Fehler einlesen, wird die *CalculateResult*-Methode aufgerufen. Diese empfängt die Daten aus dem *NCParser* und verarbeitet sie. Relevante Daten werden

zuerst in *TemporaryNCData* gespeichert. Für jedes auszuschneidende Teil, das im NC-Code beschrieben wird, erstellt die Methode einen *TCPart*. Das Datenmodell dieser Klasse ist eine Teilmenge des Informationsmodells des *NCParsers*. Nicht relevante Daten bleiben unberücksichtigt, um die Komplexität minimal zu halten. Sollten diese Daten in der Zukunft ebenfalls benötigt werden, können sie der *TCPart*-Klasse hinzugefügt werden. Neben der *PartID* zur Identifizierung des Teils, werden hauptsächlich formbeschreibende Daten erfasst. Die *RawList* beinhaltet jeden Punkt eines Blechteils, den das NC-Programm anfährt. Zusätzlich wird durch den *LaserState* innerhalb des *TCPolyRaw* definiert, ob der schneidende Laser an diesem Punkt ein- oder ausgeschaltet wird. Diese Daten in Verbindung mit Rotation und Translation können genutzt werden, um die Position, die Ausrichtung und die Umrisse eines Blechteils zu ermitteln. Weitere Daten wie zum Beispiel *HasBeenAdded* sind Daten, die die Form eines Blechteils nicht direkt betreffen. *HasBeenAdded* gibt an, ob ein bestimmtes Teil überhaupt von der Maschine gefräst wird.

Sobald alle *TCPart*-Objekte erfolgreich in die *TemporaryNCData* geschrieben wurden, wird die *PersistentNCData* Liste überschrieben. Ab diesem Zeitpunkt kann ein *DataModule* die *GetResult*-Methode nutzen, um auf Daten zuzugreifen. Die *TemporaryNCData* wird geleert und wird bis zum nächsten Aufruf der *Start*-Methode nicht mehr genutzt.

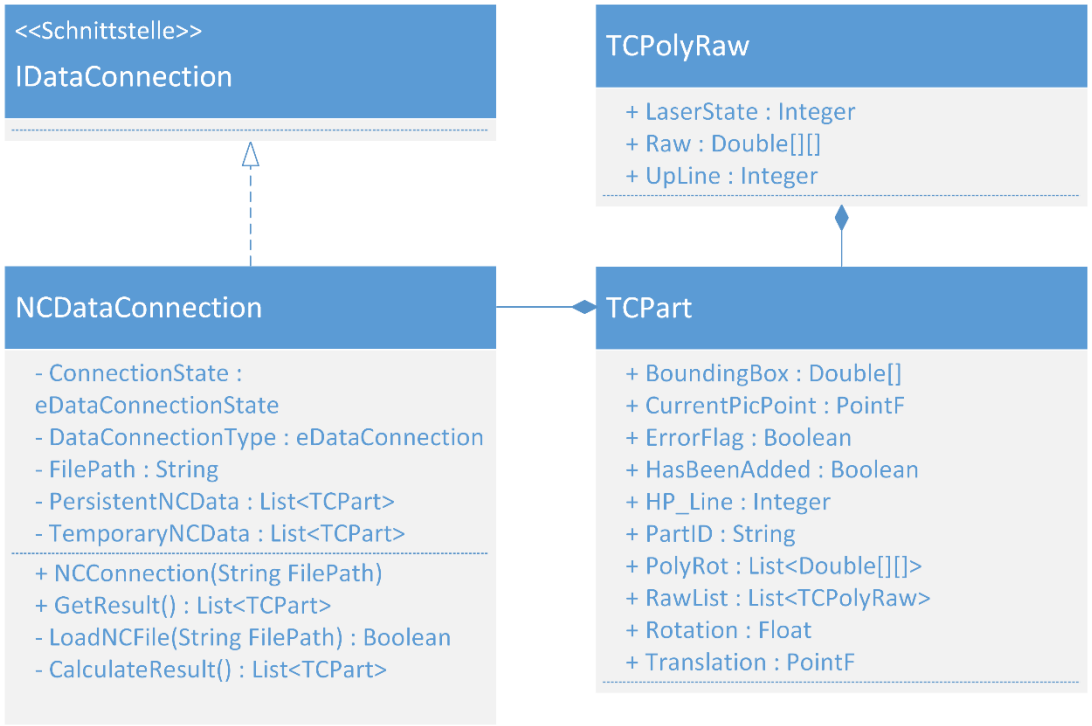

ABBILDUNG 37 – UML-KLASSENDIAGRAMM FÜR DIE NCDATACONNECTION

6.2.1.2 ProjectDataConnection

Die *ProjectDataConnection* liest Daten aus einem Dokument mit textueller Beschreibung von Projekten ein. Damit dieser Vorgang funktionieren kann, muss das Textdokument eine definierte Struktur besitzen. Zukünftige Industrie 4.0 Dokumente sollen eine solche Struktur besitzen. Da es zu dieser Zeit jedoch noch keinen Industrie 4.0 Dokumentenstandard gibt, wurde eine eigene Textdatei angelegt. Diese Datei enthält Beispieldaten für mehrere, nicht reale Projekte. Die *DataConnection* liest diese Daten ein und erstellt daraus ein eigenes Datenmodell, das wiederum an *DataModules*

angeboten wird. Zum Einlesen der Daten wird keine zusätzliche Technologie verwendet. Ein einfacher *StreamReader* aus dem *System.IO* Namensraum des .NET Frameworks wird verwendet, um die Datei einzulesen. Den Aufbau der *ProjectDataConnection* kann dem UML-Klassendiagramm aus Abbildung 38 entnommen werden.

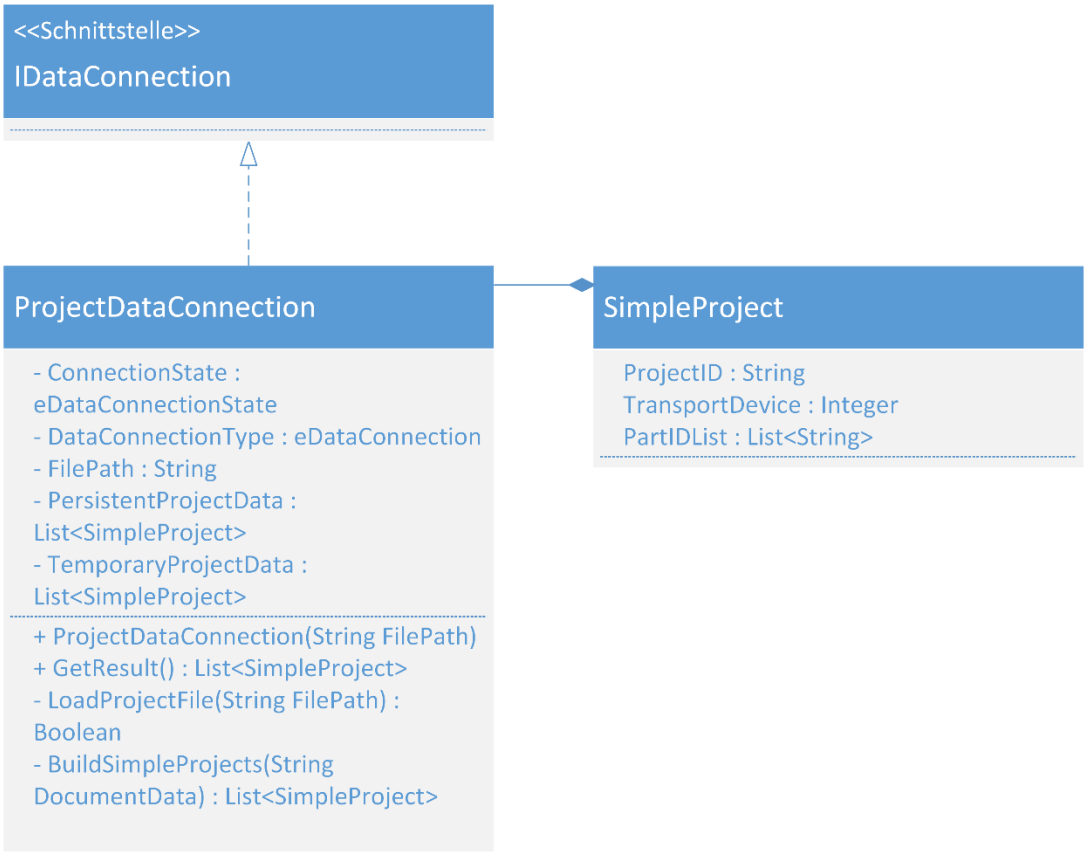

ABBILDUNG 38 – UML-KLASSENDIAGRAMM FÜR DIE PROJECTDATACONNECTION

Analog zur *NCDataConnection* implementiert die *ProjectDataConnection* die *IDataConnection*-Schnittstelle. Auch in Abbildung 38 werden die Methoden der Schnittstelle aus Gründen der Übersicht nicht gezeigt. Abbildung 35 in Kapitel 5.3.4 enthält alle Methoden der Schnittstelle inklusive einer Erklärung ihrer Funktion innerhalb des AR-Frameworks. *ConnectionState* und *DataConnectionType* sind durch die Visual Studio Vorlage für eine *DataConnection* vorgeschrieben. Der *FilePath* beinhaltet den Pfad zur Projektdatei. Zwei Listen, die *PersistentProjectData* und die *TemporaryProjectData* enthalten die Daten, die der Projektdatei entnommen wurden. Im Konstruktor der *DataConnection* wird der Pfad zur Projektdatei in Form eines Strings übergeben. Die *GetResult*-Methode wird von verknüpften *DataModules* genutzt, um auf die *PersistentProjectData* zuzugreifen. Die *private*-Methode *LoadProjectFile* wird beim Aufrufen der *Start*-Methode aus der *IDataConnection*-Schnittstelle ausgeführt. Innerhalb der Methode wird ein *StreamReader* angelegt, der den Text der Projektdatei einliest. Konnte dieser Text erfolgreich eingelesen werden, wird er in der *BuildSimpleProject*-Methode verarbeitet. In dieser Methode werden mehrere *SimpleProject*-Objekte erstellt. Für jedes Projekt in der Projektdatei wird ein SimpleProject-Objekt erstellt. Ein *SimpleProject* enthält drei Variablen, die notwendig für die Funktionalität der AR Anwendung sind. Die *ProjectID* ist ein eindeutiger Name in Form eines *Strings*, der das Projekt identifiziert. Das *TransportDevice* ist eine Nummer, die dem Projekt ein Transportmittel zuweist. Die *PartIDList* enthält eine oder mehrere

Bezeichnungen zur eindeutigen Identifizierung von Projekten. Diese Liste wird verwendet, um Projektinformationen mit den Daten aus der NC-Datei in Verbindung zu bringen. Die *SimpleProject*-Objekte werden innerhalb der *BuildSimpleProjects*-Methode in die *TemporaryProjectData* gespeichert. Nach einem kompletten Durchlauf der Methode wird die *TemporaryProjectData* in die *PersistentProjectData* kopiert und anschließend geleert. Die *ProjectDataConnection* ist danach bereit, Daten über die *GetResult*-Methode bereit zu stellen.

Durch den Aufbau dieser beiden *DataConnection* Klassen wird klar, dass die Vorlage für *DataConnections* sehr viel Einfluss auf die Struktur der Klassen hat. Werden neben den vorgegebenen Variablen alle Designrichtlinien beachtet, ähneln sich die erstellen *DataConnections* sehr. Dies sorgt für eine konsistente Nutzung und auch für ein gutes Verständnis der Klassen bei der Weiterentwicklung oder Bearbeitung von alten *DataConnections*.

6.2.2 DATENMODULE

Für jede der beiden vorgestellten *DataConnections* existiert ein *DataModule*, das die Daten aus einer *DataConnection* verarbeitet. Die Daten werden aufbereitet und über eine Schnittstelle an *FunctionModules* zur Verfügung gestellt.

6.2.2.1 NCDataModule

Das *NCDataModule* verwaltet die von der *NCDataConnection* eingelesenen NC Daten. Über mehrere *public*-Methoden bietet das *DataModule* einem *FunctionModule* Zugriff auf die erhaltenen Daten. Innerhalb dieser Methoden werden die Daten bereits aufbereitet und ausgewählt an ein *FunctionModule* angeboten.

Die Struktur des *NCDataModule* deckt sich komplett mit dem in Kapitel 5.3.2 beschriebenen vorgegebenen Aufbau. Das Modul implementiert die *IDataModule*-Schnittstelle, deren Methoden in Abbildung 39 ausgeblendet wurden. Über diese Schnittstelle kann das *DataModule* initialisiert und verwaltet werden. Neben den Methoden aus der Schnittstelle implementiert das Modul Variablen und Methoden aus der AR-Framework Vorlage für *DataModules*. *ModuleID*, *ModuleType*, *ModuleState*, *ActiveDataConnection* sowie *SupportedDataConnections* sind Teil jedes *DataModules*. Die ausgeblendeten Verwaltungsmethoden der *IDataModule*-Schnittstelle bieten Zugriff auf diese *private*-Objekte. Die *TCPartList* speichert die Daten, die aus der verbundenen *DataConnection* empfangen werden.

Bei der Initialisierung über den *AccessController* bekommt das *NCDataModule* einen *ModuleIdentifier* übergeben, der das Modul einzigartig identifiziert. Durch die *Start*-Methode des *NCDataModule* wird die *GetNCDataFromDataConnection*-Methode aufgerufen. Diese überprüft, welche der unterstützten *DataConnection*-Objekte vorliegen und nutzt die, für diese Klasse definierte Logik, um die Daten aus der *DataConnection* in das *DataModule* zu transferieren. Zu diesem Zeitpunkt unterstützt das *NCDataModule* nur die *NCDataConnection*, die dieselbe Datenstruktur besitzt wie das *DataModule*. Aus diesem Grund kann das Transferieren der Daten erfolgen, ohne diese anzupassen. Weitere *public*-Methoden bieten einem *FunctionModule* Zugriff auf ausgewählte Teile der Daten. Über *GetAllTCParts* kann sich ein *FunctionModule* eine Beschreibung aller auszufräsenden Blechteile geben lassen. Möchte das *FunctionModule* jede Teilform nur einmal erhalten, kann die *GetAllTCPartsOnce*-Methode genutzt werden. Eine Repräsentation eines spezifischen Blechteils kann ein *FunctionModule* abfragen, indem es der *GetTCPart*-Methode die *PartID* eines *TCParts* übergibt. Weitere Daten, wie die Anzahl aller *TCParts* oder ein zufälliges *TCPart,* bieten zusätzlichen Komfort.

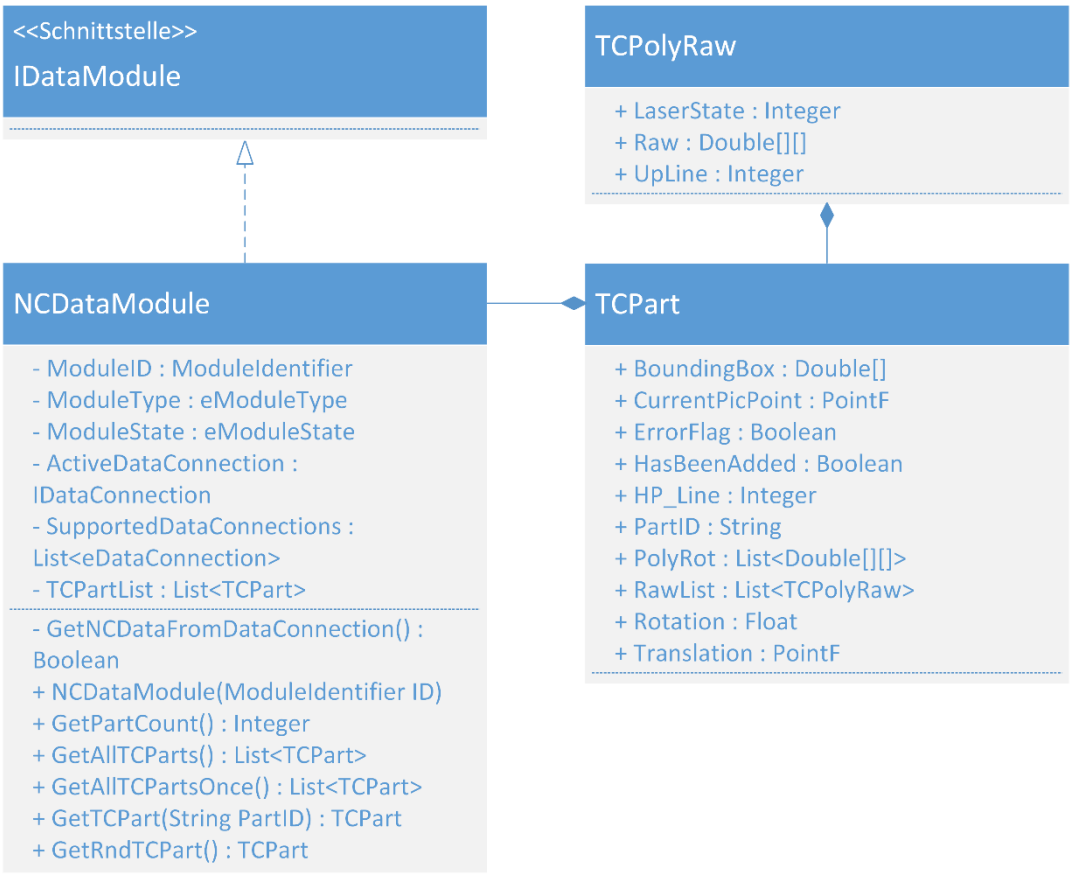

ABBILDUNG 39 - UML KLASSENDIAGRAMM FÜR DAS NCDATAMODULE

6.2.2.2 ProjectDataModule

Dieses *DataModule* greift auf Projektdaten aus der ProjectDataConnection zu und bietet ausgewählten Zugriff auf die Daten. Ein *FunctionModule* kann diesem *DataModule* gezielte Informationen aus den Projektdaten entnehmen. Wie genau diese Auswahl an *public*-Methoden funktioniert, kann dem UML-Klassendiagramm aus Abbildung 40 entnommen werden.

Wie beim *NCDataModule* implementiert das *ProjectDataModule* die *IDataModule*-Schnittstelle, um vom *AccessController* gesteuert werden zu können. Alle *private*-Variablen bis auf die *ProjectList* sind von der AR-Framework Vorlage für *DataModules* festgelegt. Die *ProjectList* beinhaltet die Projektdaten, die das *ProjectDataModule* von seinen *DataConnections* erhält. Im derzeitigen Stand besitzt das *ProjectDataModule* nur einen unterstützten *DataConnection*-Typ. Da diese *DataConnection*, die *ProjectDataConnection*, auch die *SimpleProject*-Klasse zur Datenverwaltung nutzt, muss in der *GetProjectsFromDataConnection*-Methode keine Transformation der Daten vorgenommen werden. Diese Methode wird von der *Start*-Methode beim Modulstart aufgerufen. Bevor das Modul gestartet werden kann, muss es vom *AccessController* initialisiert werden. Dieser übergibt dem Konstruktor einen eindeutigen *ModuleIdentifier*, um das *DataModule* identifizieren zu können. Ist das Modul initialisiert und die Daten von der *DataConnection* übertragen, kann ein *FunctionModule* auf die Daten zugreifen. Hierfür bereitet das *ProjectDataModule* die Daten entsprechend vor. Über die *GetProjectCount*-Methode kann die Anzahl der vorhandenen Projekte abgefragt werden. Mit Hilfe der *GetAllProjects*-Methode können alle vorhandenen *SimpleProject*-Objekte abgefragt werden. *GetProject* bietet die Möglichkeit nur ein bestimmtes Projekt zu erhalten. Die *ProjectID* in Form eines *Strings* dient zur Identifizierung. Alle Projekte, die ein bestimmtes Blechteil

enthalten, können über *GetAllProjectsIncludingTCPart* abgefragt werden. Diese *public*-Methoden unterliegen keinen semantischen Einschränkungen des AR-Frameworks. Einzig die syntaktischen Vorgaben der Vorlage für *DataModules* sind einzuhalten. Diese Vorgaben halten das *DataModule* übersichtlich und konsistent in Bezug auf andere *DataModules*.

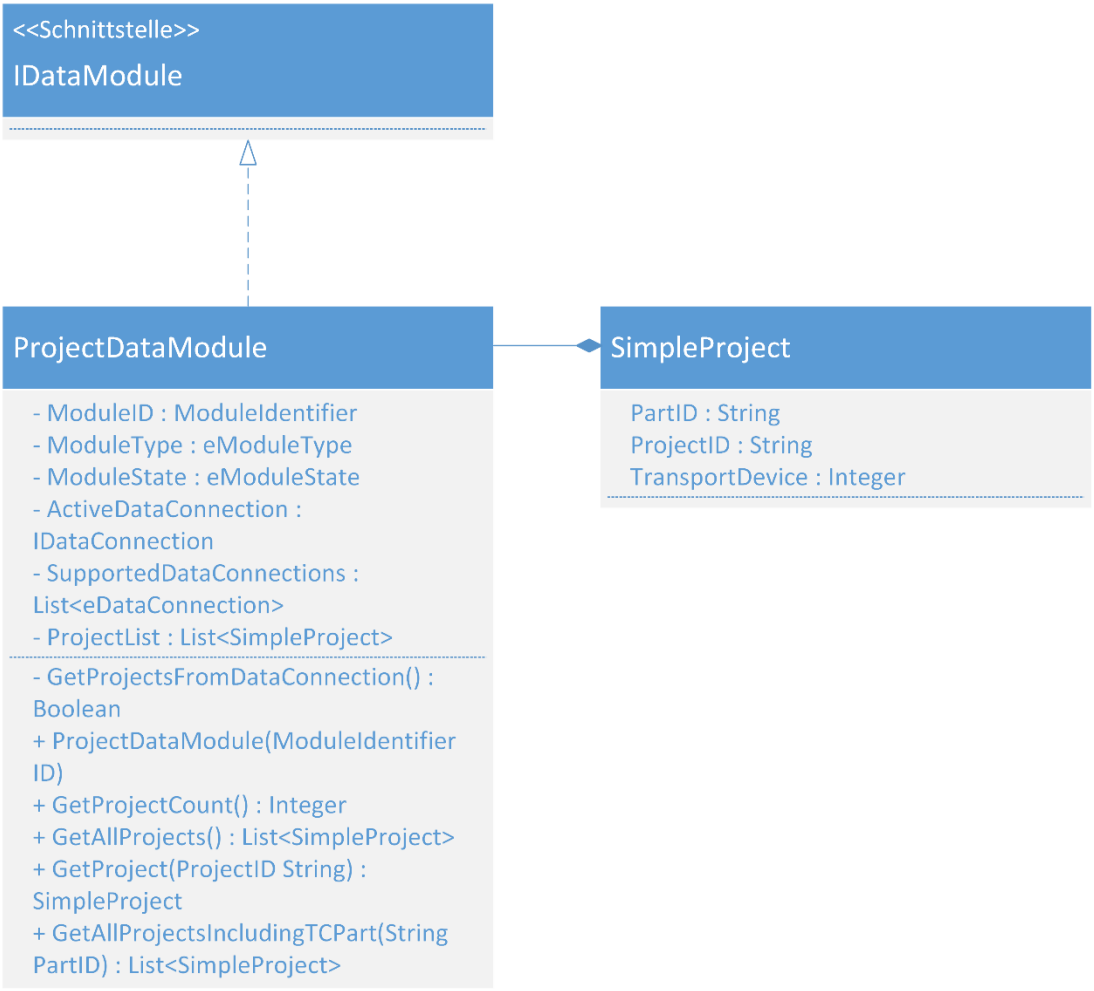

ABBILDUNG 40 - UML KLASSENDIAGRAMM FÜR DAS PROJECTDATAMODULE

6.2.3 FUNKTIONSMODULE

Um die Daten aus den angelegten *DataModules* zu verwerten, wird ein *FunctionModule* entwickelt, das beide *DataModules* nutzt. Das Modul kombiniert die Daten und bietet die erzeugten Informationen über eine Schnittstelle einem Entwickler an, der mit Hilfe der Daten eine Augmented Reality Anwendung realisieren kann.

6.2.3.1 PlateHighlightFunctionModule

Das *PlateHighlightFunctionModule* nutzt sowohl das *NCDataModule* als auch das *ProjectDataModule,* um Informationen für eine Augmented Reality Anwendung zur Verfügung zu stellen. Abbildung 41 zeigt den exakten Aufbau des *PlateHighlightFunctionModule* in Form eines UML-Klassendiagramms.

ABBILDUNG 41 - UML KLASSENDIAGRAMM FÜR DAS PLATEHIGHLIGHTFUNCTIONMODULE

Wie bei den anderen Modulen befinden sich die Verwaltungsmethoden in der zu implementierenden Schnittstelle. Im Falle eines *FunctionModules* ist dies die *IFunctionModule*-Schnittstelle. Die Methoden zur Verwaltung des *FunctionModules* sind, wie bei den restlichen UML-Diagrammen aus diesem Kapitel ausgeblendet. Die *Dependencies*-Variable wird von der AR-Framework Vorlage für *FunctionModules* vorgegeben. Für beide *DataModules*, die innerhalb dieses *FunctionModules* genutzt werden, existiert eine nicht initialisierte Variable. *NCData* und *ProjectData* werden initialisiert, sobald dem *FunctionModule* die entsprechenden *DataModules* übergeben werden. Das *ColorDict* ist ein *Dictionary* aus dem *System.Collections.Generic* Namensraum. In diesem *Dictionary* werden Projekten verschiedene Farben zugewiesen. Um die Inhalte dieses Objekts direkt in Unity nutzen zu können, verwendet das *PlateHighlightFunctionModule* die *UnityEngine.DLL*. Diese Bibliothek wird von Unity zur

Verfügung gestellt und enthält Datentypen und Methoden, die in der Unity Engine genutzt werden. Die Farben können mit Hilfe der Bibliothek im Datentyp *UnityEngine.Color* hingelegt werden. Dieser Datentyp kann in Unity Skripten direkt weiterverwendet werden, ohne ihn transformieren zu müssen.

FunctionModules nutzen den Basiskonstruktor, der im Klassendiagramm nicht angezeigt wird. *DataModules* können dem *FunctionModule* über die *SetDataModule*-Methode der *IFunctionModule*-Schnittstelle übergeben werden. Ist das Modul initialisiert und die *DataModules* übergeben, kann das *FunctionModule* zur Informationsgewinnung genutzt werden. *GetTCPartsAsDotSequence* greift auf das *NCDataModule* zurück und generiert aus den vorhandenen Daten eine Punktfolge in Form von *UnityEngine.Vector2*-Elementen. Diese Objekte repräsentieren in der Unity Engine einen zweidimensionalen Punkt innerhalb der virtuellen Landschaft. Um diese Werte zu erzeugen, wird modulintern die *CalculateActualVector2*-Methode aufgerufen. Diese berechnet aus Punktkoordinaten, der Verschiebung und der Rotation einen zweidimensionalen Vektor. Wird diese Berechnung für alle Punkte innerhalb eines *TCParts* durchgeführt, ist das Ergebnis eine Punktfolge, die die Form eines Blechteils beschreibt. Zusätzlich ist diese Form so gedreht, wie sie letztendlich aus dem Blech gefräst wird. Auch die Position innerhalb des Blechs ist in den Koordinaten enthalten, wobei diese relativ zum Punkt (0,0) der Blechkoordinaten angegeben ist. *GetTCPartsAsDotSequence* übergibt auf diese Weise alle Formen mit der entsprechenden Positionierung auf dem Blech. Möchte der Entwickler die Formen unabhängig von ihrer Position haben, kann er *GetEveryTCPartOnceAsDotSequence* oder *GetRndTCPartAsDotSequence* nutzen. Diese Methoden geben die Punktfolge für ein Blechteil zurück. Diese Punktfolge startet mit dem Punkt (0,0), da sie keine Relation zu anderen Objekten besitzt.

Die Daten aus dem *ProjectDataModule* nutzt das *FunctionModule*, um eine Liste mit allen verfügbaren Projekten zur Verfügung zu stellen. Diese Projektnamen kann der Entwickler nutzen, um über die Methode *GetTCPartsForProject* nur die Punktfolgen zu erhalten, die Blechteile mit einer bestimmten Projektzugehörigkeit beschreiben. Intern nutzt das *PlateHighlightFunctionModule* dafür die Projektdaten, um alle *PartIDs* zu einer *ProjectID* zu erhalten. Danach werden beim *NCDataModule* die *TCParts* über die *PartIDs* abgefragt. Die Punktfolgen, die bei diesem Methodenaufruf zurückgegeben werden, besitzen Koordinaten, die relativ zum (0,0) - Punkt der Blechkoordinaten stehen. Möchte der Entwickler die gleichen Punktfolgen ohne ihre exakte Position auf dem Blech haben, gibt *GetRawTCPartsForProject* die gewünschten Koordinaten zurück. In diesen Koordinaten startet jede Punktfolge am Punkt (0,0). *GetTCPartsWithProjects* ist eine Methode des *PlateHighlightFunctionModules*, die mehrere der oben genannten Methoden umfasst. Sie gibt dem Entwickler für jede Form auf dem Blech eine Punktfolge zurück. Zusätzlich zu jeder Punktfolge enthält ein *String* den Namen des zugehörigen Projekts. Durch die Nutzung aller genannten Methoden kann ein Entwickler sämtliche Formen von zu fräsenden Blechteilen in Form von zweidimensionalen Vektoren erhalten. Zusätzlich kennt er die Projektzugehörigkeit sowie die Position der Teile. Das *FunctionModule* bietet über *GetUniqueColorForProject* zusätzlich noch eine automatische Farbzuordnung von Projekten an. Die Methode unterstützt zehn definierte Farben, die einzigartig Projekten zugewiesen werden. Der Entwickler muss sich dementsprechend nicht mehr darum kümmern, für jedes Projekt eine Farbe anzulegen.

Möchte ein Entwickler nicht selbstständig Game Objekte in Unity erzeugen und die Punktfolgen zeichnen, kann auch dieser Schritt vom *FunctionModule* übernommen werden. *GetTCPartAsGameObject* und *GetAllTCPartsAsGameObject* geben dem Entwickler komplette Game Objekte zurück, die die gezeichneten Fräsformen enthalten. Diese Game Objekte kann der Entwickler mit wenigen Zeilen Code in seiner virtuellen Umgebung einfügen. Die Komplexität eine Anwendung zu

entwickeln, sinkt hierdurch erheblich, da der Entwickler nicht verstehen muss, wie er die Punktfolgen zu nutzen hat. Durch die Einbindung der *UnityEngine.DLL* in die *FunctionModules* können Modulentwickler die Komplexität bestimmen, mit der sich der Entwickler der Augmented Reality Anwendung auseinandersetzen muss. Vom Durchreichen der Daten, wie zum Beispiel über die *GetAllProjects*-Methode bis zum vollständigen Erstellen von virtuellen Elementen, wie in der *GetAllTCPartsAsGameObject*-Methode, sind beide Extreme möglich.

6.3 Ergebnis

Die Augmented Reality Anwendung, die mit Hilfe des AR-Frameworks erstellt wird, wurde in diesem Kapitel bisher nur funktionell und strukturell beschrieben. Dieser Abschnitt zeigt die einzelnen Schritte der Entwicklung bis hin zum fertigen Ergebnis mit Hilfe visueller Darstellungen.

Augmented Reality Anwendungen erweitern die reale Welt mit virtuellen Elementen. Da die reale Welt bereits vorhanden ist, wird bei der Entwicklung der Anwendung mit der virtuellen Welt begonnen. Das Ziel bei der Entwicklung der Anwendung ist es, die Informationen aus dem AR-Framework zu beziehen und in eine virtuelle Welt zu projizieren. Wenn dieser Schritt funktioniert, wird die virtuelle Welt mit der realen Welt ersetzt. Diese Vorgehensweise hat sich bei der Entwicklung von Augmented Reality Anwendungen in kleinem Umfang bewährt.

Um Informationen in die virtuelle Welt zu transportieren, muss eine virtuelle Welt existieren. Ein CAD-Modell einer TruLaser 5030 Werkzeugmaschine dient als Vorlage für eine digitale Werkzeugmaschine. Dieses CAD-Modell wird in Unity importiert und kann danach als digitales Abbild für eine echte Werkzeugmaschine genutzt werden. Damit das Werkstück in der Werkzeugmaschine besser zu sehen ist, werden große Teile der Abdeckung der Werkzeugmaschine entfernt. Abbildung 42 zeigt die Werkzeugmaschine ohne Verkleidung in einer Unity Anwendung. Auf diese Blechplatte werden in den nächsten Schritten Informationen eingeblendet.

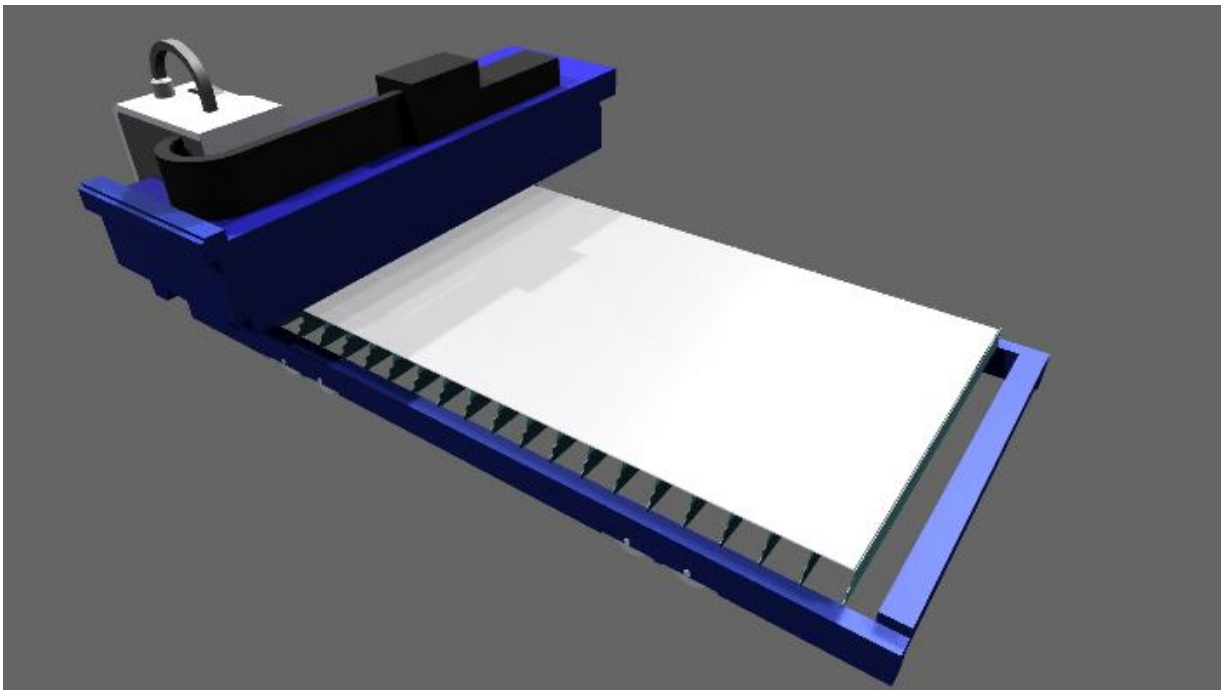

Abbildung 42 - Abbildung einer Trumpf TruLaser 5030 Werkzeugmaschine ohne Verkleidung in Unity

Zu diesem Zeitpunkt ist die digitale Welt bereit, mit Hilfe des AR-Frameworks erweitert zu werden. Dafür wird ein Unity Skript erstellt, das die in Kapitel 6.2 beschriebenen Module nutzt, um Informationen zu beziehen. Abbildung 43 zeigt einen Ausschnitt des genutzten C# Skripts aus Unity.

```
1  //Define Document Paths
2  string NCPath = @"..\\Duenn_Einfach_St_2mm.lst";
3  string ProjectPath = @"..\\Projects.txt";
4
5  // Initialize Controller and create Data Modules needed
6  ARFramework.Controller.AccessController Controller =
7      new ARFramework.Controller.AccessController();
8  ARFramework.Common.ModuleIdentifier NCModuleID =
9      Controller.InitializeDataModule(ARFramework.Common.eModuleType.NCDataModule);
10 ARFramework.Common.ModuleIdentifier ProjectModuleID =
11     Controller.InitializeDataModule(ARFramework.Common.eModuleType.ProjectDataModule);
12
13 // Create Data Connections
14 ARFramework.Connection.NCDataConnection NCConnection =
15     new ARFramework.Connection.NCConnection(NCPath);
16 ARFramework.Connection.ProjectDataConnection ProjectConnection =
17     new ARFramework.Connection.NCConnection(ProjectPath);
18
19 // Connect Data Modules and Data Connections
20 Controller.SetDataConnection(NCModuleID, NCConnection);
21 Controller.SetDataConnection(ProjectModuleID, ProjectConnection);
22
23 // Start Data Modules, which causes Data Connections to also start
24 Controller.StartDataModule(NCModuleID);
25 Controller.StartDataModule(ProjectModuleID);
26
27 // Get PlateHighlightFunctionModule
28 ARFramework.Functions.PlateHighlightFunctionModule HighlightFunctionModule =
29     Controller.PlateHighlightFunctionModule(
30     new List<ARFramework.Common.ModuleIdentifier>()
31     { NCModuleID, ProjectModuleID });
32
33 //Use Function Module
34 System.Collections.Generic.List<List<Vector2>> AllDotSequencesv2 =
35 HighlightFunctionModule.GetListOfDotSequences();
```

ABBILDUNG 43 - NUTZUNG DES AR-FRAMEWORKS IN EINEM UNITY C# SKRIPT

In diesem Ausschnitt wird das AR-Framework initial eingerichtet. In Zeile 2 und 3 werden Pfade zu den zwei einzulesenden Dateien angelegt. In Zeile 6 wird der *AccessController* initialisiert, um in den folgenden Zeilen die nötigen *DataModules* anzulegen. Anschließend werden die *DataConnections* erstellt. Der Pfad zu dem jeweiligen Dokument wird als Parameter übergeben. *DataModule* und *DataConnection* werden in Zeile 20 und 21 miteinander verknüpft. Das *DataModule* wird hierbei durch seinen *ModuleIdentifier* vertreten, da der Entwickler nicht direkt auf ein *DataModule* zugreifen kann. Die *DataModules* besitzen nun eine DataConnection und können in den Zeilen 24 und 25 gestartet werden. Der Start der *DataModules* führt automatisch zum Start der hinterlegten DataConnections. In Zeile 28ff wird das *PlateHighlightFunctionModule* initialisiert. Vorausgesetzt die *DataModules* wurden erfolgreich gestartet und sind in einem aktiven Zustand, übergibt der Controller das *FunctionModule* an den Entwickler. Dieser kann nun auf die *public*-Methoden des *FunctionModules* zugreifen. In Zeile 34f greift der Entwickler auf die *GetListOfDotSequence*-Methode zu. Dieser Zugriff ist für eine Implementierung, in welcher der Entwickler die Punktfolgen selbstständig nutzt, um Game Objekte in Unity zu erzeugen. Um zu zeigen wie sehr das AR-Framework die Komplexität senken kann, wird in den folgenden Codezeilen die Methode *GetAllTCPartsAsGameObject* verwendet. Diese Methode nimmt es dem Entwickler ab, die Punktfolgen mit Hilfe von eigenem Code in Unity zu zeichnen. Stattdessen ist in Abbildung 44 zu sehen, dass der Entwickler mit einem Aufruf in Zeile 42 vorgefertigte Game Objekte

bekommt, die relativ zueinander positioniert sind. Alles was der Entwickler noch machen muss, ist diese Game Objekte in seine virtuelle Welt platzieren. Er kann die Game Objekte entweder direkt in die virtuelle Welt setzen oder er knüpft die Game Objekte an ein bereits vorhandenes Game Objekt an. In Abbildung 44 werden die erhaltenen Game Objekte an das Werkstück geheftet, das sich bereits in der virtuellen Welt befindet. Da *GetAllTCPartsAsGameObject* eine Liste an Objekten zurückgibt, ist eine *foreach*-Schleife nötig, um die Objekte zuzuordnen. Wenn nur ein Game Objekt zurückgegeben werden würde, wäre lediglich eine Zeile Code nötig, um ein Game Objekt in der virtuellen Welt anzuzeigen.

```
41  // Use Function Module to get shapes of TCParts as Game Objects without Color
42  List<GameObject> GameObjectList = HighlightFunctionModule.GetAllNCPartsAsGameObject(false);
43
44  // Find Workpiece in virtual model to place the shapes
45  GameObject workpiece = GameObject.Find("TruelaserWorkpiece");
46  foreach (GameObject shape in GameObjectList)
47  {
48      // Relate shapre to workpiece
49      shape.transform.parent = workpiece.transform;
50  }
```

ABBILDUNG 44 - DIREKTER ZUGRIFF AUF GAME OBJEKTE ÜBER DAS AR-FRAMEWORK IN EINEM UNITY SKRIPT

Abbildung 45 zeigt das Ergebnis des Arbeitsschrittes. Die vom NCDataModule bereitgestellten Formen und Positionen der auszufräsenden Blechteile werden vom *PlateHighlightFunctionModule* aufbereitet und als Game Objekte direkt an Unity übergeben.

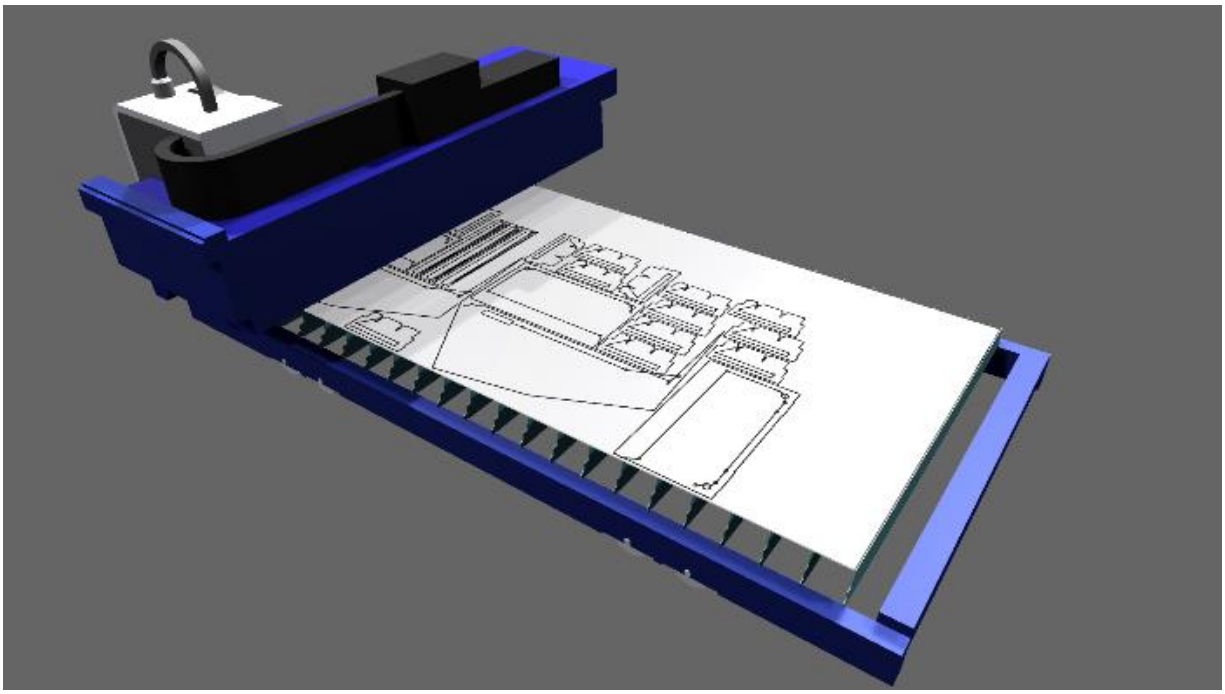

ABBILDUNG 45 - EINGEBLENDETE INFORMATIONEN AUS EINEM NC PROGRAMM AUF EINER TRUMPF TRULASER 5030 WERKZEUGMASCHINE OHNE VERKLEIDUNG

Als nächster Schritt wird neben dem *NCDataModule* auch das *ProjectDataModule* mit einbezogen. Dafür sollen die einzelnen Formen farblich gekennzeichnet werden. Das *PlateHighlightFunctionModule* kombiniert hierfür beide *DataModules* und nutzt seine interne Zuordnung von Farben und Projekten. Das Ergebnis ist in Abbildung 46 sichtbar. Um dieses Ergebnis zu produzieren, muss einzig der

Parameter beim Methodenaufruf von *GetAllTCPartsAsGameObject* von *false* auf *true* gesetzt werden. Dieser Boolean gibt an, ob die Elemente farblich differenziert werden sollen oder nicht. Die beispielhafte Projektdatei ordnet jedem Projekt eine bestimmte Form der Blechteile zu. Teile mit der gleichen Form sind entsprechend in der gleichen Farbe abgebildet. Gleiche Formen könnten auch unterschiedlichen Projekten angehören. Auch dieser Anwendungsfall könnte dargestellt werden.

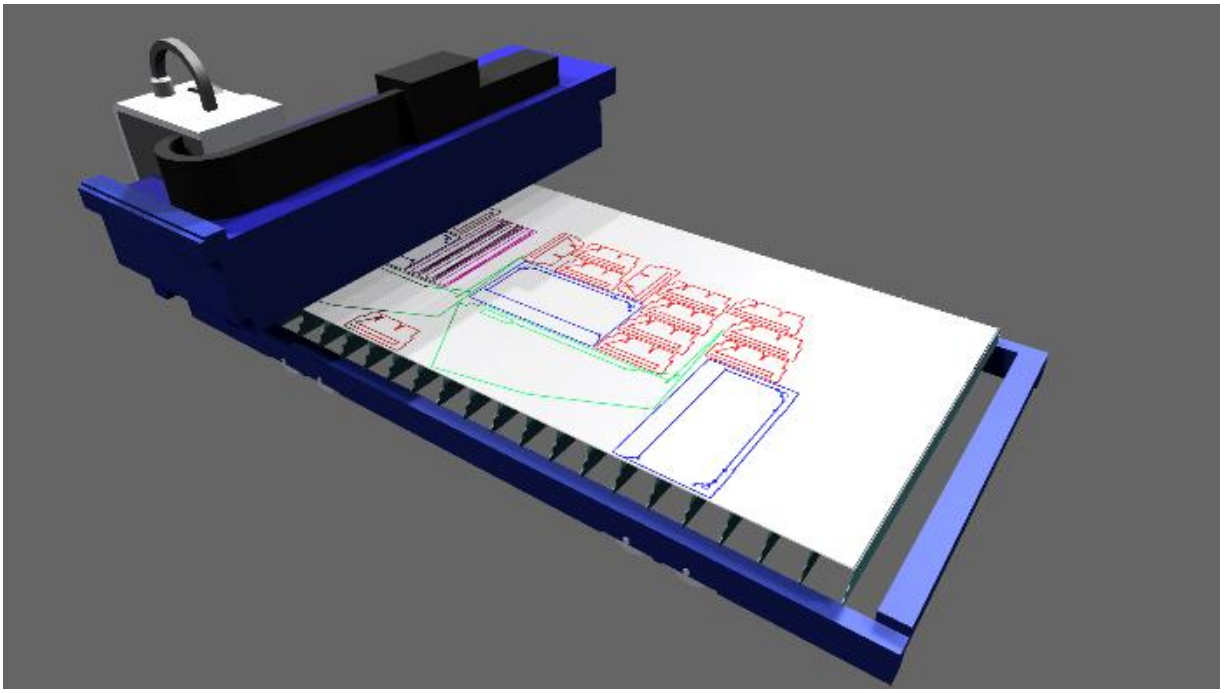

ABBILDUNG 46 - MIT ZUSÄTZLICHEN INFORMATIONEN AUSGESTATTETE ABBILDUNG EINES NC PROGRAMM AUF EINER TRUMPF TRULASER 5030 WERKZEUGMASCHINE OHNE VERKLEIDUNG

Bis hierher wurde gezeigt, dass das AR-Framework in der Lage ist Daten zu erfassen, zu kombinieren und virtuelle Elemente über eine Schnittstelle an Unity zu übertragen. Der nächste Schritt ist das Entfernen der virtuellen Welt. Die virtuellen Elemente, in diesem Fall die farbigen Formen der Bleichteile, sollen in der realen Welt auf eine Blechplatte projiziert werden. Für diesen Schritt wird das ARToolKit in Unity eingebunden. Das Toolkit bietet mehrere Skripte, die an Game Objekte gehängt werden können. Abbildung 47 zeigt den strukturellen Aufbau der Anwendung und die verwendeten Skripte in einem baumartigen Modell.

Das *PlateHighlightApplication*-Objekt ist die Wurzel der Baumstruktur. Dieses Game Objekt hat keine visuelle Repräsentation und dient lediglich zur Verwaltung der anderen Objekte. Da ein Game Objekt Zugriff auf alle unterliegenden Elemente hat, bietet es sich an, Verwaltungsskripte wie den *ARController* oder das *ARFramework*-Skript mit diesem Game Objekt zu verknüpfen. Das *ARFramework*-Skript beinhaltet die komplette Nutzung des AR-Frameworks. In diesem Skript werden, wie in den Quellcode-Ausschnitten aus Abbildung 43 und 44 zu sehen, Daten in das Framework geladen, Informationen produziert und Game Objekte vom AR-Framework bezogen, um sie anzuzeigen. Um zu wissen, an welcher Position die Game Objekte im Bild angezeigt werden sollen, wird das *ARMarker*-Skript genutzt. Dieses Skript ist Teil des ARToolKit. Es repräsentiert einen Marker in der realen Welt. Entsprechend ist es wichtig, diesem Skript einen Marker zuzuweisen. In diesem Fall wird der Hiro-Marker des ARToolKits benutzt. Der Marker wird neben dem Werkstück platziert und dient als Ankerpunkt für die virtuellen Elemente. Neben dem Markertyp wird dem Skript ein eindeutiger Name zugewiesen, ein sogenannter *Tag*. Der Grund für die Zuweisung wird später erklärt.

Das *ARController*-Skript, ein weiteres Skript des ARToolKits, verwaltet das AR-Tracking. Es trennt die reale Welt und die virtuellen Elemente in zwei Ebenen. Um die reale Welt zu erfassen, nutzt der *ARController* das Kamerabild und platziert es in einer Hintergrundebene. Die virtuellen Elemente werden in einer Ebene angezeigt, die weiter im Vordergrund liegt. Dieses Vorgehen ist nötig, um zu gewährleisten, dass die virtuelle Welt die reale Welt überlagern kann. Das unterliegende Game Objekt *SceneRoot* ist, wie das *PlateHighlightApplication*-Objekt kein visuell sichtbares Game Objekt. Dieses Objekt wird an eine feste Stelle innerhalb der Szene in Unity positioniert. In diesem Fall befindet sich das Objekt am Punkt (0, 0, 0) im globalen Koordinatensystem der Unity Anwendung.

Das *AROrigin*-Skript verwaltet alle definierten *ARMarker*. Beim Start der Anwendung sucht das Skript sämtliche angelegten Marker und speichert diese in einer Liste. Warum dieses Vorgehen wichtig ist, wird mit der Erklärung der beiden folgenden Game Objekte erläutert.

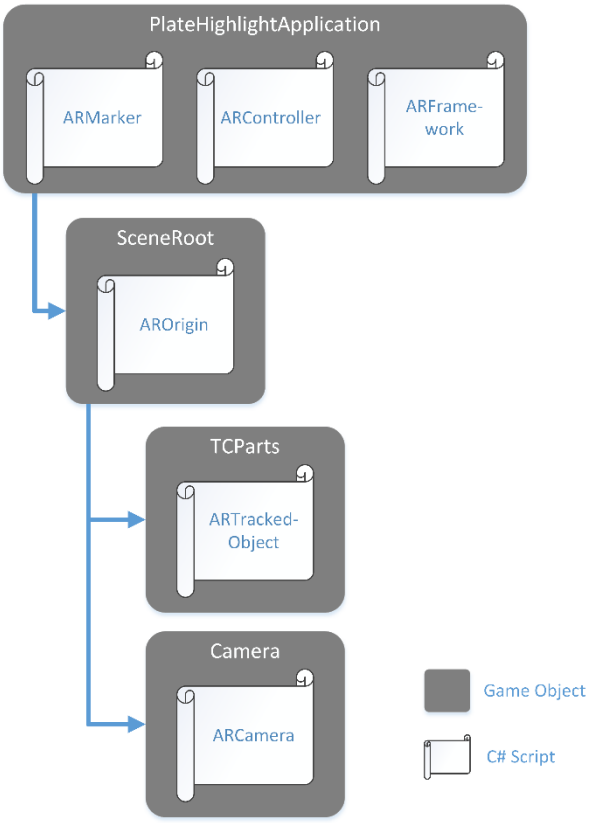

ABBILDUNG 47 - UNITY STRUKTUR DER MIT HILFE DES AR-FRAMEWORKS ERSTELLTEN AR ANWENDUNG

Das *ARCamera*-Skript im *Camera*-Objekt benutzt das *AROrigin*-Skript, um an sämtliche *ARMarker* zu gelangen. Ein *ARMarker* besitzt einen *Boolean*, der aussagt, ob ein entsprechender Marker im Bildausschnitt gefunden wurde oder nicht. Das *ARCamera*-Skript nutzt diese Information sowie die Koordinaten des gefundenen Markers und berechnet die Stelle des Markers innerhalb des Bilds aus dem Sichtwinkel der genutzten Kamera. An dieser Position soll ein virtuelles Element angezeigt werden.

Das *ARTrackedObject*-Skript repräsentiert dieses Element. Das Game Objekt, das mit dem Skript verknüpft ist, wird auf dem Kamerabild angezeigt. In diesem Fall ist dieses Game Objekt das *TCParts*-Objekt. Die vom AR-Framework erhaltenen Game Objekte werden innerhalb des *ARFramework*-Skripts dem *TCParts*-Objekt angehängt. Bewegt sich ein Game Objekt in Unity, bewegen sich automatisch die

untergeordneten Game Objekte mit. Diese Eigenschaft wird genutzt, um die virtuellen Formen der Frästeile zu bewegen, falls sich das Werkstück und somit der Marker bewegt. Um die Verbindung zwischen dem *TCParts*-Objekt und dem *ARMarker* herzustellen, wird der bereits erwähnte *Tag* des *ARMarker*-Skripts genutzt. Das *ARTrackedObject*-Skript besitzt ebenfalls einen *Tag*. Das *AROrigin*-Skript kombiniert *ARTrackedObjects* und *ARMarker* anhand dieses *Tags* und kann so jedes *ARTrackedObject* an der passenden Stelle anzeigen.

Die Augmented Reality Anwendung, die durch dieses Unity Projekt generiert wird, benötigt kein virtuelles Werkstück mehr, sondern kann die reale Welt mit den virtuellen Umrissen der Frästeile erweitern. Die Abbildungen 48 und 49 zeigen eine Blechplatte, die mit einem Hiro-Marker versehen wurde. Auf dieser Platte wird farblich angezeigt, welche gefrästen Formen welchem Projekt zugeordnet sind. Aufgenommen wurden beiden Bilder in einer realen DMG-Werkzeugmaschine.

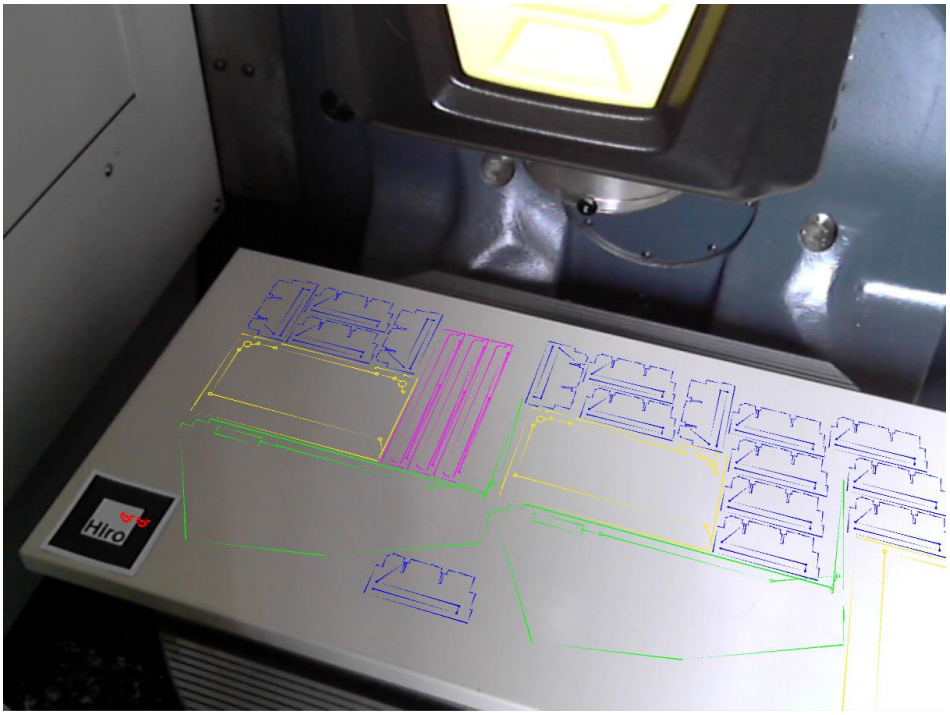

ABBILDUNG 48 - DARSTELLUNG VON BERECHNETEN INFORMATIONEN DES AR-FRAMEWORKS IN EINER AR-ANWENDUNG

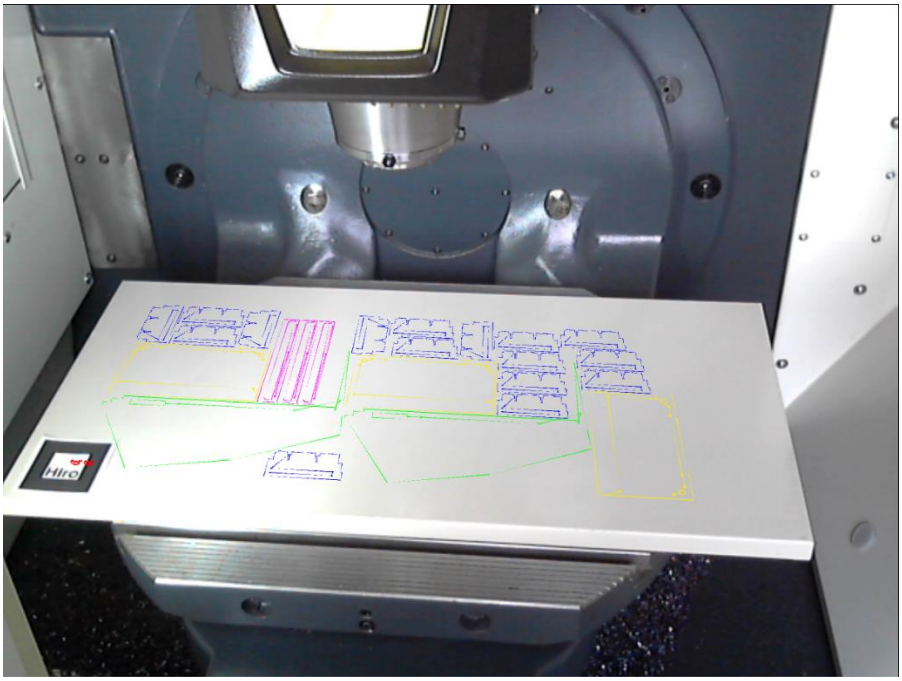

ABBILDUNG 49 – WEITERE DARSTELLUNG VON BERECHNETEN INFORMATIONEN DES AR-FRAMEWORKS IN EINER AR-ANWENDUNG

7 EVALUATION

Nachdem durch die Erstellung einer Beispielanwendung aus Kapitel 6 gezeigt wurde, dass das AR-Framework funktioniert, wird in diesem Kapitel aufgezeigt, dass sämtliche Anforderungen an das Framework erfüllt wurden. Hierfür wird für jede funktionale Anforderung aus Kapitel 1.3 nachgewiesen, dass die gewünschte Funktionalität umgesetzt ist. Anschließend werden nichtfunktionale Anforderungen an das Framework betrachtet und festgelegt, ob diese ebenfalls erfüllt wurden. Alle Anforderungen sind im folgenden Text enthalten und durch die Anforderungsnummer eindeutig den Anforderungen aus Kapitel 1.3 zugewiesen. Eine Abbildung für beide Anforderungsarten gibt eine Zusammenfassung über die erfüllten und nicht erfüllten Anforderungen.

7.1 EVALUATION ANHAND DER FUNKTIONALEN ANFORDERUNGEN

Die Anforderung PFA10 verlangt vom AR-Framework die Möglichkeit, auf verschiedene Datenquellen zugreifen zu können. Das Prinzip der Verbindungsmodule ermöglicht es dem Framework diese Anforderung zu erfüllen. In der Beispielanwendung aus Kapitel 6 werden NC-Code und textuelle Daten eingelesen. Auch eine Nutzung von Datenbanken, Cloud Speicher oder direkte Verbindungen zu einer Maschine sind über dieses Konstrukt möglich. Funktionsmodule können mehrere dieser Datenmodule nutzen, um deren Daten zu kombinieren und dadurch einen Mehrwert zu generieren. Diese Anforderung wurde in PFA11 gestellt. In PFA12 wird vom Framework verlangt, den Entwickler einer Augmented Reality Anwendung zu unterstützen. Neben der Datenerfassung und der Datenverarbeitung ist dies die dritte Kernfunktion des AR-Frameworks. Die direkte Anbindung an Unity und die Möglichkeit, virtuelle Objekte innerhalb des AR-Frameworks zu erstellen und an Unity zu übergeben, senkt die Komplexität bei der Entwicklung von Augmented Reality Anwendungen. Die Bereitstellung von Vorlagen zur Erweiterung des Frameworks wurde in PFA13 gefordert. Die Visual Studio Vorlagen für Datenmodule, Funktionsmodule und Verbindungsmodule ermöglichen eine einfache Erweiterung und erfüllen somit die Anforderung. Eine ausdrückliche Anforderung, festgehalten in PFA14 besagt, dass Unity für die Erstellung von Augmented Reality Anwendungen genutzt werden muss. Durch die Integration der *UnityEngine.DLL* in das AR-Framework und der Nutzung des ARToolKits für Unity, kann auch diese Anforderung erfüllt werden. In PFA15 wird der Wunsch nach Umsetzungen von möglichen Datenmodulen geäußert. In der Anwendung aus Kapitel 6 wird ein Datenmodul für NC Programme sowie ein Datenmodul für ein projektbeschreibendes Dokument genutzt. Diese Module stehen im Framework bereit. Neben den Datenmodulen wird in dieser Anwendung ein Funktionsmodul zum Erzeugen von farbigen Formen auf Basis von NC-Programmen genutzt. Auch dieses Funktionsmodul ist im Umfang des Frameworks enthalten. Mit diesem Modul ist der Wunsch nach vorgefertigten Funktionsmodulen aus PFA16 ebenfalls erfüllt. In Abbildung 50 ist zu sehen, dass alle gestellten Anforderungen erfüllt wurden. Die optionale Anforderung PFA15 beinhaltet weitere Datenarten, die bisher nicht in Form eines Datenmoduls umgesetzt wurden. Aus diesem Grund ist diese Anforderung nur zur Hälfte erfüllt.

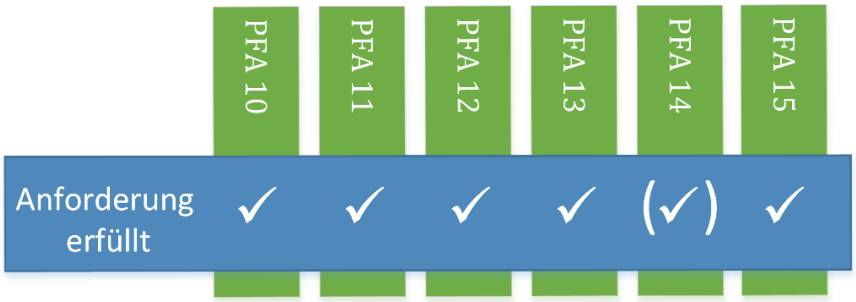

ABBILDUNG 50 - ZUSAMMENFASSUNG ÜBER DIE ERFÜLLUNG DER FUNKTIONALEN ANFORDERUNGEN AN DAS AR-FRAMEWORK

7.2 EVALUATION ANHAND DER NICHTFUNKTIONALEN ANFORDERUNGEN

Die Kapselung von Funktionalitäten des AR-Frameworks in verschiedene Module wurde in PNA10 gefordert. Die Designentscheidung, die Funktionalität des AR-Frameworks in Datenmodule, Funktionsmodule und Verbindungsmodule zu trennen, trägt zur Erfüllung dieser Anforderung bei. Zusätzlich ist jedes der drei Funktionsgebiete sehr modular aufgebaut. Ein Modul kann unabhängig von anderen Modulen seines Typs agieren. Einzig der Zugriffsmanager verbindet die Module zu einem System. PNA11 verlangt einen einfachen Umgang ohne steile Lernkurve bei der Nutzung des Frameworks. Hierbei soll vor allem Wert auf die Konsistenz innerhalb des Frameworks geachtet werden. Um diese Anforderung zu erfüllen, sind im AR-Framework Richtlinien und Ratschläge aus dem Buch „*FRAMEWORK DESIGN GUIDELINES*“ von Cwalina und Abrams [11] umgesetzt worden. Dieses Buch wurde von Microsoft als offizielle Vorlage für .NET-Frameworks gewählt. Wie in den Klassendiagrammen der Anwendung aus Kapitel 6 sichtbar wird, führen die Ratschläge der Autoren zu einem in sich konsistenten und einfach zu verstehenden Framework. Durch die Nutzung dieses Buches wird gleichzeitig PNA12 erfüllt. In dieser Anforderung wird definiert, dass bei der Umsetzung des Frameworks auf bewährte Richtlinien gesetzt werden soll. Nach PNA 13 müssen Anwendungen, die mit Hilfe des AR-Frameworks erstellt werden, auf unterschiedlichen Zielplattformen ausführbar sein. Durch die Nutzung von Unity sind alle genannten Zielplattformen abgedeckt. Die Art der Technologie des AR-Frameworks wird in PNA14 festgelegt. Das AR-Framework muss als Bibliothek vorliegen. Diese Anforderung erleichtert die Arbeit mit Unity, da Unity in der Lage ist, C# Bibliotheken über das Mono Framework direkt einzubinden und zu nutzen. Das AR-Framework nutzt diese Funktionalität von Unity und erfüllt somit die Anforderung. Die letzte Anforderung an das AR-Framework ist die Umsetzung einer prototypischen Anwendung durch die Nutzung des Frameworks, beschrieben in PNA15. In Kapitel 6 wird ausführlich beschrieben, wie im Laufe dieser Arbeit eine Anwendung mit Hilfe des AR-Frameworks entstanden ist. Diese Anforderung ist dadurch ebenfalls erfüllt. Einen Überblick über alle nicht funktionalen Anforderungen kann Abbildung 51 entnommen werden.

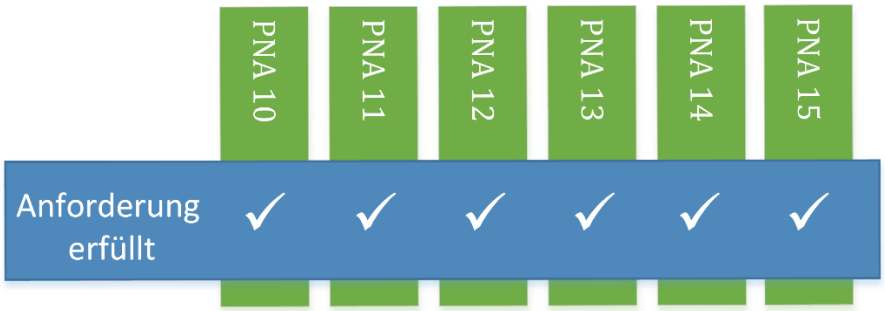

ABBILDUNG 51 - ZUSAMMENFASSUNG ÜBER DIE ERFÜLLUNG DER NICHT FUNKTIONALEN ANFORDERUNGEN AN DAS AR-FRAMEWORK

Wie in den Abbildungen 50 und 51 zu erkennen ist, sind alle Anforderungen an das AR-Framework erfüllt. Einzig PFA14 ist nur zur Hälfte erfüllt, da diese Anforderung einige potentielle Datenmodule aufzählt. Zum einen ist diese Anforderung als Wunschkriterium definiert, zum anderen dienen die verschiedenen Datentypen nur als Vorschlag. Der umklammerte Haken in Abbildung 50 soll dennoch zeigen, dass nicht alle der vorgeschlagenen Datenmodule umgesetzt wurden. Da ansonsten alle Anforderungen vollkommen erfüllt sind, kann das AR-Framework in Zukunft für die Erstellung weiterer Augmented Reality Anwendungen genutzt werden.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Framework entwickelt, das in der Lage ist, Daten aus unterschiedlichen Quellen zu erfassen. Die erfassten Daten können von Mehrwertfunktionen genutzt werden, um aus Rohdaten produktionsrelevante Informationen zu gewinnen. Diese Informationen werden genutzt, um einen Entwickler bei der Erstellung von Augmented Reality Anwendungen für die Industrie zu unterstützen. Die Unterstützung kann auf zwei verschiedene Arten erfolgen. Zum einen kann der Entwickler Informationen aus dem AR-Framework beziehen, die er in seiner Augmented Reality Anwendung selbst verarbeiten kann. Diese Informationen liegen in Unity-konformen Datenarten vor. Dadurch muss der Entwickler die Daten nicht mehr transformieren, sondern kann sie direkt verwenden. Möchte der Entwickler sich andererseits nicht selbst mit den Informationen befassen, bietet das AR-Framework die Möglichkeit, komplette virtuelle Elemente innerhalb des AR-Frameworks zu erzeugen und an Unity zu übergeben. Der Entwickler muss anschließend diese Elemente nicht mehr erstellen, sondern nur noch platzieren.

Vor der Entwicklung des AR-Frameworks befasste sich die Arbeit mit den nötigen Grundlagen. Diese Grundlagen umfassen alle für diese Arbeit relevanten Bereiche. Zur Einordnung welche Bereiche relevant sind, wurde der Titel der Arbeit verwendet. Entsprechend des Titels wurden Grundlagen zu Augmented Reality, Industrie 4.0, der Entwicklung von Anwendungen und Richtlinien zu Frameworks erläutert. Nachfolgend wurde der Stand der Technik erforscht. Aktuelle Augmented Reality Anwendungen in der Industrie haben gezeigt, welche Möglichkeiten es derzeit gibt und welche Probleme noch bestehen. Ein Vergleich verschiedener Software Development Kits zum Entwickeln von Augmented Reality Anwendungen war ausschlaggebend für die Wahl des ARToolKits als SDK für die Augmented Reality Komponente des AR-Frameworks. Im letzten Teil des Stands der Technik wurden verschiedene Möglichkeiten der Datenerfassung untersucht, mit dem Ziel eine Technologie zu finden, die unterschiedliche Arten von Daten programmatisch zur Verfügung stellen kann.

Nach der Entwicklung des Frameworks wurde mit einem Industriepartner eine Anwendung entwickelt, die NC-Programme einliest und mit Projektdokumenten kombiniert, um einen Mehrwert zu bilden. Diese Mehrwertinformationen ermöglichten eine farbige Hervorhebung von gefrästen Blechteilen, geordnet nach Projektzugehörigkeit. Diese Umsetzung einer Anwendung sowie eine Evaluation des AR-Frameworks auf Basis der gestellten Anforderungen zeigen, dass das Framework für die Erstellung weiterer Augmented Reality Anwendungen geeignet ist.

Trotz der Erfüllung sämtlicher Kriterien und der erfolgreichen Entwicklung einer Anwendung, gibt es Möglichkeiten das AR-Framework zu verbessern. Die komplette Funktionalität des AR-Frameworks in eine Bibliothek zu komprimieren, die von Unity eingelesen werden kann, birgt Einschränkungen. Technologien, die direkt im AR-Framework genutzt werden, müssen mit dem .NET Framework 3.5 oder niedriger kompatibel sein. Das aktuelle .NET Framework 4.5 sowie die ältere Version 4.0 werden von Unity nicht unterstützt. Dadurch können diese Bibliotheken nicht in einer Unity Anwendung genutzt werden. Ein anderes Problem zeigt sich, wenn mehrere Technologien innerhalb des Frameworks genutzt werden. Diese Technologien benötigen Ressourcen, um Daten zu erfassen und zu verarbeiten. Die *DataConnections* und *DataModules* brauchen bei größeren Datenmengen ebenfalls zusätzliche Ressourcen. Die *FunctionModule*s benötigen mehr Rechenkraft bei der Analyse größerer Datenmengen. Auch die Erstellung von Game Objekten innerhalb des AR-Frameworks benötigt einige Ressourcen. Bei vielen Game Objekten erreicht ein herkömmlicher Computer schnell seine Grenzen. Auf Smartphones wird diese Grenze offensichtlich noch früher erreicht.

Eine Lösung für diese Probleme wäre die Auslagerung der einzelnen Module in eine Cloud Umgebung. Der modulare Aufbau des AR-Frameworks legt die solide Grundlage, um das Framework in die Cloud zu verschieben. Neben einer besseren Skalierung durch die Umgehung der leistungslimitierenden Faktoren, wäre der Einsatz weiterer Technologien möglich. Das AR-Framework wäre nicht mehr nur auf Bibliotheken im .NET Raum beschränkt. Eine Cloud Lösung würde jedoch nicht nur Vorteile mit sich bringen. Zusätzliche Kosten, eine deutlich kompliziertere Kommunikation zwischen den Modulen untereinander sowie zwischen Zugangsmanager und Modulen und der Zwang mit dem Internet verbunden zu sein, sind potentielle negative Punkte.

Um das AR-Framework tiefgründiger zu testen und zu analysieren, sind zudem weitere Modul-Implementierungen nötig. Ohne weitere Module ist das AR-Framework nur eingeschränkt nützlich, da die Wiederverwertungskomponente fehlt. Aus diesem Grund werden nach Abschluss dieser Arbeit neue Module entwickelt, um das AR-Framework produktiv einsetzen zu können. Ein potentieller Anhaltspunkt sind die noch fehlenden Datenmodule aus den optionalen Anforderungen.

9 LITERATURVERZEICHNIS

[1] T. Bauernhansl, M. ten Hompel und B. Vogel-Heuser, Industrie 4.0 in Produktion, Automatisierung und Logistik, Springer Fachmedien Wiesbaden, 2014.

[2] VDI Statusreport, Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0), 2015.

[3] P. Milgram, A TAXONOMY OF MIXED REALITY VISUAL DISPLAYS, IEICE Transactions on Information Systems, Vol E77-D, No.12, 1994.

[4] S. You, U. Neumann und R. Azuma, Hybrid Inertial and Vision Tracking for Augmented Reality Registration, Virtual Reality, 1999. Proceedings., IEEE, 2002.

[5] E. Bostanci, N. Kanwal, S. Ehsan und A. Clark, User Tracking Methods for Augmented Reality, International Journal of Computer Theory and Engineering, Vol. 5, No. 1, 2013.

[6] A. Cheok und Y. Li, „Ubiquitous interaction with positioning and navigation using a novel light sensor-based information transmission system," in Personal and Ubiquitous Computing, vol. 12, pp. 445-458, 2008.

[7] A. Golding und N. Lesh, „Indoor navigation using a diverse set of cheap, wearable sensors," in Third International Symposium on Wearable Computers, pp. 29–36, 1999.

[8] „Unity," Unity Technologies, [Online]. Available: https://store.unity.com/products/unity-personal?_ga=1.40406690.1401125302.1459761555. [Zugriff am 20 September 2016].

[9] „Gamesindustry.biz," Gamesindustry.biz, 30 Juni 2015. [Online]. Available: http://www.gamesindustry.biz/articles/2015-06-30-vr-is-going-to-yield-this-staggering-orgasm-of-new. [Zugriff am 20 September 2016].

[10] „Unity3D," Unity Technologies, [Online]. Available: https://unity3d.com/partners/microsoft/hololens. [Zugriff am 20 September 2016].

[11] K. Cwalina und B. Abrams, Framework Design Guidelines, Second Edition, Pearson Education, 2008.

[12] „Framework Entwurfsrichtlinien," Microsoft , Juli 2016. [Online]. Available: https://msdn.microsoft.com/de-de/library/ms229042(v=vs.110).aspx. [Zugriff am September 2016].

[13] B. Myers and G. Stylos, "Improving API Usability," Communication of the ACM, pp. 62-69, Juni 2016.

[14] „Emerging Technology Hype Cycle for 2016," Forbes, [Online]. Available: http://blogs-images.forbes.com/gartnergroup/files/2016/08/Emerging-Technology-Hype-Cycle-for-2016_Infographic_no-url-01-Forbes-1200x950.png. [Zugriff am 20 September 2016].

[15] F. Echtler, F. Sturm, K. Kindermann, G. Klinker, J. Stilla, J. Trilk und H. Najafi, „The Intelligent Welding Gun: Augmented Reality for Experimental Vehicle Construction,"

Virtual and Augmented Reality Applications in Manufacturing, Springer London, pp. 333--360, 2004.

[16] K. Pentenrieder, C. Bade, F. Doil und P. Meier, „Augmented Reality-based factory planning - an application tailored to industrial needs," Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium , pp. 31-42, 2007.

[17] H. Regenbrecht, G. Baratoff und W. Wilke, „Augmented reality projects in the automotive and aerospace industries," Proc. COMPUT GRAPH, IEEE, 2005.

[18] W. Wohlgemuth und G. Triebfürst, „ARVIKA: Augmented Reality for Development, Production and Service," Proceedings of DARE 2000 on Designing Augmented Reality Environments, pp. 151--152, 2000.

[19] J. Zhou, I. Lee, B. Thomas, R. Menassa, A. Farrant und A. Sansome, „In-Situ Support for Automotive Manufacturing Using Spatial Augmented Reality," The International Journal of Virtual Reality, 2012.

[20] J. Serván, F. Mas, J. Menéndez und J. Ríos, „Using augmented reality in AIRBUS A400M shop floor assembly work instructions," AIP Conference Proceedings, 1431, , pp. 633-640, 2012.

[21] L. Garza, G. Pantoja, P. Ramírez, H. Ramírez und N. Rodríguez, „Augmented Reality Application for the Maintenance of a Flapper Valve of a Fuller," Procedia Computer Science, Volume 25, pp. 154-160, 2013, .

[22] M. Fiorentino, A. Uva, M. Gattullo, S. Debernardis und G. Monno, „Augmented reality on large screen for interactive maintenance instructions," Computers in Industry, Volume 65, Issue 2, pp. 270-278, Februar 2014.

[23] S. Webel, U. Bockholt, T. Engelke, N. Gavish, M. Olbrich und C. Preusche, „An augmented reality training platform for assembly and maintenance skills," Robotics and Autonomous Systems 61, p. 398–403, 2013.

[24] A. Syberfeldt, O. Danielsson, M. Holm und L. Wang, „Dynamic operator instructions based on augmented reality and rule-based expert systems," 48th CIRP Conference on MANUFACTURING SYSTEMS, 2015.

[25] J. Azpiazu, S. Siltanen, P. Multanen, A. Mäkiranta, N. Barrena, A. Díez, J. Agirre und T. Smith, „Remote support for maintenance tasks by the use of Augmented Reality: the ManuVAR project," CARVI 2011: IX Congress on virtual reality applications, Alava, Spain,, November 2011.

[26] A. Doshi, R. Smith, B. Thomas und C. Bouras, Use of projector based augmented reality to improve manual spot-welding precision and accuracy for automotive manufacturing, The International Journal of Advanced Manufacturing Technology, 2016.

[27] J. Zhang, S. Ong und A. Nee, „Development of an AR system achieving in situ machining simulation on a 3-axis CNC machine," Comp. Anim. Virtual Worlds 2010, p. 103–115.

[28] J. Torok, M. Kocisko, M. Teliskova und M. Janak, „Increasing of the Work Productivity of CMM Machine by Applying of Augmented Reality Technology," MATEC Web of Conferences 68, 2016.

[29] P. Sommer, A. Atmosudiro, J. Schlechtendahl, A. Lechler und A. Verl, iWindow – Concept of an intelligent window for machine tools using augmented reality, 2015.

[30] A. Nee, S. Ong, G. Chryssolouris und D. Mourtzis, „Augmented reality applications in design and manufacturing," CIRP Annals - Manufacturing Technology 61, p. 657–679, 2012.

[31] N. J. Andre, A Modular Approach to the Development of Interactive Augmented Reality Applications, Electronic Thesis and Dissertation Repository. Paper 1800, 2013.

[32] H. Ramirez, E. Mendivil, P. Flores und M. Gonzalez, „Authoring Software for Augmented Reality applications for the use of maintenance and training process," Procedia Computer Science 25, p. 189 – 193, 2013.

[33] C. Kollatsch, M. Schumann, P. Klimant, V. Wittstock und M. Putz, „Mobile Augmented Reality Based Monitoring of Assembly Lines," Procedia CIRP, Volume 23, pp. 246-251, 2014.

[34] D. Amin und S. Govilkar, „Comparative Study of Augmented Reality SDK's," International Journal on Computational Sciences & Applications (IJCSA), Februar 2015.

[35] I. Marneanu, M. Ebner und T. Rößler, Evaluation of Augmented Reality Frameworks, International Journal of Innovation Management, 2014.

[36] „PTC.de," PTC Inc., [Online]. Available: http://www.ptc-de.com/presse/2015/ptc-to-acquire-vuforia-from-qualcomm. [Zugriff am 20 September 2016].

[37] R. Nikhil, T. Nikhil, K. Sukrit, S. Hari und G. Prasad, A Survey on Text Mining and Sentiment Analysis for Unstructured Web Data, International Journal of Emerging Technologies and Innovative Research , 2015.

[38] A. McCallum und K. Nigam, A Comparison of Event Models for Naive Bayes Text Classification, AAAI-98 workshop on learning for text categorization. Vol. 752, 1998.

[39] T. Joachims, „Support Vector Learning," in Text categorization with Support Vector Machines: Learning with many relevant features, Springer Berlin Heidelberg, 2005, pp. 137-142.

[40] S. Panichella, „Textual Analysis or Natural Language Parsing? A Software Engineering Perspective," University of Zurich, 2015.

[41] M. Strljic, Entwicklung eines Datenhaltungskonzepts für Industrie 4.0 Komponenten, Institut für Steuerungstechnik, Universität Stuttgart, 2016.

[42] M. Telenta, L. Kos, R. Akers und L. I, „A link between CAD models and physics codes," in 43rd EPS Conference on Plasma Physics, 2016.

[43] A. Daffner und W. Jürgensen, Anbindung von Web Frontends und Web Services mittels SAP. Net Connector an ein ERP System, 2008.

[44] O. Winkler, M. Valas, O. P und L. Landryofa, „Development of Tools for Accessing Data Retrieved from MES Software Applications," 2009.

[45] D. SHOJAEI, A. RAJABIFARD, M. Kalantari, I. D. Bishop und A. Aien, Development of a 3D ePlan/LandXML visualisation system in Australia, Third International FIG Workshop on 3D Cadastres: Developments and Practices, 2012.

[46] W. Lee, D. Gao und J. Cheung, „An NC tool path translator for virtual machining of precision optical products," Journal of Materials Processing Technology Volume 140, Issues 1–3,, p. 211–216, 2003.

[47] International Electrotechnical Commission, IEC International Standard IEC 61131-3 - Programmable Controllers, Part 3: Programming Language, IEC, 2003.

[48] D. Darvas, I. Majzik und E. Vinuela, Formal Verification of Safety PLC Based Control Software, Integrated Formal Methods: 12th International Conference, 2016.

[49] W. Mahnke, S.-H. Leitner und M. Damm, OPC Unified Architecture, Springer-Verlag Berlin Heidelberg 2009, 2009.

[50] „OPC UA SDK Übersicht," OPC Foundation, [Online]. Available: https://www.unified-automation.com/de/produkte/sdk-uebersicht.html. [Zugriff am 20 September 2016].

[51] H. Junker, „IT-Sicherheit für Industrie 4.0 und IoT," Datenschutz und Datensicherheit - DuD, 2015.

10 ABBILDUNGSVERZEICHNIS

11 TABELLENVERZEICHNIS

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Unterschrift:

Stuttgart, 28.09.2016

Declaration

I hereby declare that the work presented in this thesis is entirely my own.
I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.
Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.
The electronic copy is consistent with all submitted copies.

Signature:

Stuttgart, 28.09.2016