

Konzeption und prototypische Umsetzung
einer Steuerzentrale eines smarten Büros mit
dem Fokus einer einfachen Handhabung der
formalisierten Interaktionen für
Softwareentwickler

MASTER-THESIS

für die Prüfung zum
Master of Science
des Studienganges Professional Software Engineering

an der

Knowledge Foundation @ Reutlingen University

von

Mikka Jenne

Abgabedatum 31. August 2022

Bearbeitungszeitraum	24 Wochen
Teilnehmernummer	800864
Kurs	PSEJG20
ErstprüferIn	Prof. Dr. Natividad Martinez Madrid
ZweitprüferIn	Dr. Robin Braun



Zusammenfassung

Smart Home sowie Smart Office, die detaillierte Ausprägung des Smart Home, sind Unterrubriken des Internet of Things (IoT). Diese fokussieren die Vernetzung aller Arten von technischen Geräten und die Realisierung verschiedenster Automatisierungsverfahren und Prozessabbildungen. Durch die Vielzahl an Möglichkeiten zur Erfassung von Automationen, Regeln und Geräten kann der Input der Anwender, bzw. Entwickler rasant ansteigen. Es ist dem Anwender meist kaum gestattet mit den vorhandenen Softwarelösungen eine inhaltlich individuelle Anpassung der Regeldefinitionen vorzunehmen.

Diese Master-Thesis befasst sich mit der Konzeption und prototypischen Umsetzung eines Frameworks einer Steuerzentrale im Rahmen eines intelligenten Büros mit dem Fokus auf eine einfache Handhabung der formalisierten Interaktionen für Softwareentwickler. Das System bietet eine klare Struktur zur Definition von Regeln, sodass individuelle Anwendungsfälle in wenigen Schritten umgesetzt werden können. Die Komplexität dieser ist jedoch abhängig von dem zu realisierenden Prozess und kann durch die formalisierten Interaktionen selbst nicht reduziert werden.

Um die Forschungsfrage im Rahmen dieser Thesis zu beantworten sowie das Ziel dieser Arbeit zu erreichen, werden mehrere Forschungsmethoden angewendet. Es wird ein systematisches Literaturreview durchgeführt, das den aktuellen Stand der Technik in Bezug auf die Forschungsfrage dieser Arbeit eruiert. Anschließend wird eine Zielgruppenanalyse erstellt, sowie Anwendungsfälle entwickelt und Experteninterviews geleitet, wodurch Anforderungen an das System extrahiert und identifiziert werden. Mithilfe der erhobenen Informationen und gewonnenen Ergebnissen wird das Framework konzipiert und prototypisch implementiert. Damit der entstandene Prototyp auf die Usability testbar ist und die Anforderungen als auch die Forschungsfrage evaluiert werden können, werden nach der Implementierung Usability-Tests und abschließende Experteninterviews veranlasst. Diese helfen bei der Evaluation.

Abstract

Smart home and smart office, the detailed form of the smart home, are sub-categories of the Internet of Things (IoT). These focus on the networking of all types of technical devices and the implementation of a wide variety of automation processes and process mapping. Due to the large number of options for recording automations, rules and devices, the input from users and developers can increase rapidly. In most cases, the user is not allowed to adapt the content of the rule definitions individually with the existing software solutions.

This master's thesis deals with the conception and prototypical implementation of a system for a control center within the framework of a intelligent office with the focus on easy handling of formalized interactions for software developers. The system offers a clear structure for defining rules so that individual use cases can be implemented in just a few steps. However, the complexity of these depends on the process to be implemented and cannot be reduced by the formalized interactions themselves.

In order to answer the research question in this thesis and to achieve the goal of this work, several research methods are used. A systematic literature review is carried out, which determines the current state of the art in relation to the research question of this work. A target group analysis is created, use cases are developed and expert interviews are conducted, whereby requirements for the system are extracted and identified. With the help of the information collected and the results obtained, the framework is designed and implemented as a prototype. So that the resulting prototype can be tested for usability and the requirements as well as the research question can be evaluated, usability tests and concluding expert interviews are caused after implementation. These help with the evaluation.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Forschungsfrage	4
1.3 Zielsetzung der Arbeit	4
1.4 Forschungsstrategie und Forschungsmethoden	4
1.4.1 Experteninterview	5
1.4.2 Systematisches Literaturreview	5
1.4.3 Usability-Test	5
1.5 Aufbau der Arbeit	6
2 Grundlagen	8
2.1 Internet der Dinge	8
2.1.1 Paradigmen und Kommunikationsmodelle	13
2.1.2 Historische Entwicklung	14
2.1.3 Ziele von IoT	15
2.2 Smart Home	16
2.2.1 Smart Office - Intelligentes Büro	20
2.2.2 Historische Entwicklung	21
2.2.3 Ziele von Smart Home	22
2.3 Technologien	23
2.3.1 Übertragungsmethoden	23
2.3.2 MQTT	24
2.3.3 AMQP	27
2.4 Home Assistant	28
2.4.1 Konzept und Architektur	29
2.4.2 Ziele und Schwerpunkte	31
2.4.3 Stärken und Schwächen	32
2.4.4 Optionen der Regel- und Automatisierungserstellung	33
2.5 openHAB	34
2.5.1 Konzept und Architektur	35
2.5.2 Ziele und Schwerpunkte	39
2.5.3 Stärken und Schwächen	39

2.5.4	Optionen der Regel- und Automatisierungserstellung	40
2.6	Vergleich von Home Assistant und openHAB	41
3	Stand der Technik	42
3.1	Systematisches Literaturreview	42
3.1.1	Ziele des Systematischen Literaturreviews	42
3.1.2	Suchstrategie- und anfragen	42
3.1.3	Datenextraktion und Synthese	45
3.2	Zusammenfassung	45
3.2.1	Publikationen	46
3.2.2	Stand der Technik aus Nutzer- und Produktsicht	48
4	Anforderungsanalyse	50
4.1	Marktanalyse	50
4.1.1	Allgemeine Marktsituation und Marktprognose	51
4.2	Zielgruppenanalyse	55
4.2.1	Ziel der Zielgruppenanalyse	55
4.2.2	Zielgruppendefinition	55
4.2.3	Zielgruppe	56
4.3	Anwendungsfälle - Use Cases	58
4.3.1	Check-in mit einem Service-Roboter	59
4.3.2	Notfall-Evakuierung mit einem Service-Roboter	63
4.4	Experteninterview	66
4.4.1	Ziel des Experteninterviews	66
4.4.2	Aufbau des Experteninterviews	66
4.4.3	Zusammenfassung der Experteninterviews	67
4.5	Anforderungen	67
5	Konzeption	70
5.1	Ziel der Konzeption	70
5.2	Anwendungsumfeld	71
5.3	Konzept	71
5.3.1	Konzeptkomponenten	72
5.3.2	Sichtweisen	73
5.4	Architekturkonzept	77
5.4.1	Aufbau des Frameworks	79
5.4.2	Kommunikationsschicht	80
5.4.3	Logik- und Prozessschicht	82
6	Umsetzung	84
6.1	Auswahl des Frameworks	84
6.1.1	OSGi	85
6.1.2	Spring	85

6.2	Implementierung	86
6.2.1	Generische Programmierung	86
6.2.2	Zustandsraum	87
6.2.3	Reflection	89
6.2.4	Transformation	89
6.2.5	Inverse Transformation	93
6.2.6	Regeldefinition	94
6.2.7	Parallelisierung	95
6.2.8	Regelwerk -und management	96
6.2.9	Implementierung von Anwendungsfällen	96
6.3	Fazit der prototypischen Implementierung	99
6.3.1	Modellierungsvorgaben und -grenzen	100
7	Evaluation	102
7.1	Usability-Test	102
7.1.1	Ziel des Usability-Tests	102
7.1.2	Durchführung des Usability-Tests	102
7.1.3	Fazit	104
7.2	Experteninterview	104
7.2.1	Ziele des Experteninterviews	105
7.2.2	Fazit	105
7.3	Evaluation der Anforderungen	106
7.4	Evaluation der Forschungsfrage	110
8	Fazit	111
9	Ausblick	113
Literaturverzeichnis		VI
Anhang		VI
A		VI
B		VII
C		X
D		XI
E		XV
F		XXI
G		XXIII

Kapitel 1

Einleitung

Die folgende Master-Thesis befasst sich mit der Konzepterstellung einer Steuerzentrale, die dem Entwickler die formalen Interaktionen, wie z.B. weitere Funktionen hinzuzufügen, erleichtern sollen. Hierfür werden bereits bestehende Plattformen für Smart Home analysiert und daraus ein Konzept erstellt, das den Anforderungen entsprechend das Implementieren von weiteren Anwendungsfällen einfach handhaben lässt.

In diesem Teil der Arbeit wird auf die Motivation des Themas eingegangen. Darüber hinaus werden sowohl die Forschungsfragen als auch die Zielsetzung der Arbeit genauestens dargelegt. Darauf folgend findet eine Übersicht über die Arbeit im Gesamten statt, mit der die Inhalte angerissen werden. Eine nähere Betrachtung des Standes der Technik untermauert die Beweggründe für diese Themenwahl und ihre Ausarbeitung.

1.1 Motivation

Jede neu entwickelte Technologie durchlebt im Laufe der Entstehung und Publikation ein enormes Aufsehen, so lange bis diese Technik eine standardisierte Verwendung in der Gesellschaft findet oder sich als unpraktikabel erweist und nicht weiter vorangetrieben oder eingestellt wird. Zu Beginn des Aufkommens der Idee und der Forschung wird viel darüber fantasiert, debattiert und geplant, ohne jedoch die Ausmaße, Resultate und Umsetzungen abwegen zu können. Durch fehlende Erfahrungen und nicht ausgereifte Konzepte werden Höhepunkte und Illusionen erwartet, die zu diesem Zeitpunkt technisch womöglich nicht umsetzbar sind. Solche Versprechungen und Übertreibungen, sogenannte Hypes, durchlaufen bestimmte Phasen in ihrer Entwicklung, wonach sich eine aussichtsreiche technologische Idee von einer wirtschaftlich nicht umsetzbaren differenziert.

Die oben erwähnten Phasen der Entwicklung sind in einem sogenannten Hype-Zyklus, engl. Hype-Cycle [GARTNER 2022], dargestellt. Dieser Zyklus ist ein visualisiertes Modell für den Verlauf der Entwicklung einer neuen Technologie von der Innovation und Entstehung bis hin zur Forschung, Umsetzung und zur ausgereiften Marktfähigkeit.

Erfunden wurde der Hype Cycle von der Gartner Inc. Forschungsgruppe. Die Definitionen der

Entwicklungsphasen¹ wurden durch die Mitarbeiterin Jackie Finn geprägt. Diese sind wie folgt in fünf Phasen untergliedert:

1. *Innovationsauslöser, engl. Innovation Trigger*: Ein potentieller technologischer Durchbruch löst die Dinge aus. Frühe Proof-of-Concept (PoC) Ansätze und ein großes Medieninteresse lösen eine erhebliche Publizität aus. Oft gibt es keine brauchbaren Produkte und die Marktreife ist nicht bewiesen. [GARTNER 2022]
2. *Höhepunkt überhöhter Erwartungen, engl. Peak of Inflated Expectations*: Frühe Publizität bringt eine Reihe von Erfolgsgeschichten hervor – oft begleitet von zahlreichen Misserfolgen. Einige Unternehmen ergreifen Maßnahmen; viele nicht. [GARTNER 2022]
3. *Trog der Ernüchterung, engl. Trough of Disillusionment*: Das Interesse schwindet, wenn Experimente und Implementierungen die Erwartungen nicht erfüllen. Hersteller der Technologie schaffen den Durchbruch oder scheitern. Investitionen werden nur fortgesetzt, wenn die übriggebliebenen Anbieter ihre Produkte zur Zufriedenheit der Kunden verbessern. [GARTNER 2022]
4. *Steigung der Erleuchtung, engl. Slope of Enlightenment*: Mehr Beispiele dafür, wie die Technologie dem Unternehmen zugute kommen kann, beginnen sich herauszukristallisieren und werden allgemeiner verstanden. Produkte der zweiten und dritten Generation erscheinen von den Technologieanbietern. Mehr Unternehmen finanzieren Pilotprojekte; konservative Unternehmen bleiben vorsichtig. [GARTNER 2022]
5. *Plateau der Produktivität, engl. Plateau of Productivity*: Mainstream-Akzeptanz beginnt sich abzuheben. Kriterien zur Bewertung der Rentabilität des Anbieters sind klarer definiert. Die breite Markteinsetzbarkeit und Relevanz der Technologie zahlen sich eindeutig aus. [GARTNER 2022]

Nachdem ein innovativer Gedanke, z.B. die Vollautomatisierung eines Gebäudes oder eines uningeschränkt interagierenden Service-Roboters, den *Höhepunkt überhöhter Erwartungen* passiert hat, folgt der *Trog der Ernüchterung*. In Folge dessen wird festgestellt, dass die Erwartungen zum aktuellen Zeitpunkt nicht vollständig, bzw. nur zu einem geminderten Teil in die Realität umgesetzt werden können und dadurch an Interesse verlieren. Wird die Technologie erneut aufgegriffen, findet eine realitätsbezogene Beurteilung der Innovation statt, die dazu beitragen kann, dass die Technologie wieder an Interesse gewinnt. Die objektive und realitätsnahe Betrachtungsweise kann ein neues und realistischeres Bild der Potentiale, aber auch der Grenzen aufzeigen. Mit dem neu gewonnenen Maßstab kann die ehemals neue innovative Idee in eine routinierte Technologie übergehen, die eventuell an Anerkennung gewinnt und in der breiten Masse akzeptiert wird. Die Technologie erfährt mit steigender Zuwendung eine stetigere Weiterentwicklung, die dann zu einer Community geformt wird. Mit der Erreichung dieses Status befindet sich die Innovation, bezogen auf den Hype Cycle, in der letzten Phase, dem *Plateau der Produktivität*, und bestätigt so die Marktreife. Dieser Zeitpunkt löst die Zukunftsvision auf und es handelt sich um eine am Markt etablierte Technologie.

¹Die Entwicklungsphasen der Gartner Inc. ist unter folgender URL zu finden: "<https://www.gartner.com/en/research/methodologies/gartner-hype-cycle>"

Zum aktuellen Zeitpunkt befindet sich die Technologie rund um Plattformen für intelligente Geräte im privaten als auch im unternehmerischen Bereich, engl. *Smart Home (SH)* oder *Connected Home*, im Anfangsstadium der letzten Phase, dem sogenannten *Plateau der Produktivität*. Mit zunehmender Akzeptanz werden im Umfeld des Internets der Dinge, engl. *Internet of Things*, stetig Szenarien entwickelt, die das Wachstum und die Verwendung von solchen Plattformen vorantreibt. Mit einer immer tiefer gehenden Forschung und Umsetzung von Anwendungsbeispielen werden Bereiche eröffnet, die eine solche Plattform im privaten als auch im geschäftlichen Umfeld immer attraktiver werden lässt. Mit steigender Konnektivität und Kompatibilität mehrerer Geräte und Gegenstände können Szenarien und Prozessautomatisierungen, wie die Steuerung von Service-Robotern, umgesetzt werden. Der jetzige technologische Fortschritt und die über die Forschungsjahre gesammelten Erfahrungen bringen das Segment der intelligenten Geräte der IoT den ursprünglich angedachten Visionen und Ideen näher, sodass ein weiterer Ausbau dieser Technologie und dessen Anwendung stattfindet und sich vollständig in den Markt etabliert. Der finale Schritt der endgültigen Marktreife ist ein faszinierender und wichtiger Beweggrund für meine Motivation, mich dieser Technologie und der dahinterstehenden Theorie zu widmen. Ein weiterer dazu beitragender Aspekt ist die Möglichkeit der Steuerung und Verknüpfung von Geräten sowie die Automatisierung von Prozessen innerhalb eines intelligenten Büros. Stetig hinzukommende Anwendungsfälle abzudecken und einfach implementieren zu können, einen Ansatz zu schaffen, der einem Softwareentwickler formalisierte Interaktionen ermöglicht und trotzdem Handlungsspielraum für Prozessabbildungen lässt, sind zusätzliche Impulse, warum ich mich dieser Thematik zuwende.

Die Einsatzgebiete von Kompaktlösungen, Plattformen und Steuereinheiten, beziehungsweise von Gateways und intelligenten Geräten zielen räumlich auf Gebäude, Häuser und Wohnungen ab und bieten viele Verwendungs- und Einsatzmöglichkeiten, die stetig ausgebaut und optimiert werden. Diese sehen wie folgt aus:

- Komfort
- Entertainment
- Überwachung und Sicherheit
- Steuerung von Prozessen
- Management von Automatisierungen (Automationen)

Neben der Affinität von Smart Home zum Internet der Dinge und der damit einhergehenden Technologie bringt diese Vorteile mit sich, wie z.B. die Modernisierung von Wohn- und Bürogebäuden und die routinierte Ausführung und Abarbeitung von Prozessen, die dem Nutzer Aufgaben und Arbeiten abnehmen.

1.2 Forschungsfrage

Die in der Arbeit zentral behandelte Forschungsfrage ist wie folgt definiert:

F: Wie kann man die Usability von Prozessautomatisierungen und Regeldefinitionen von SmartHome-Plattformen optimieren, sodass die formalisierten Interaktionen für den Softwareentwickler einfacher in der Handhabung sind?

1.3 Zielsetzung der Arbeit

Das Ziel dieser Thesis ist die Erarbeitung eines Konzeptes und die prototypische Implementierung des Frameworks, welches dem Softwareentwickler das Umsetzen von Regeln, sowie zu automatisierende Prozesse im Rahmen der Programmierung erleichtert, jedoch in der Ausprägung der Regeln nicht einschränkt. Dabei wird sich an bestehenden Produkten orientiert, die versuchen, diese Automatisierungen für den Nutzer vereinfacht darzustellen. Voraussetzung ist trotz allem eine intensive Auseinandersetzung mit der Materie. Mit dem Framework soll dem Anwender eine Struktur an die Hand gegeben werden, mit der Regeln und Prozesse innerhalb eines smarten Büros einfach abgebildet werden können. Die Abarbeitung von Regeln und Prozessen sollen über das Framework koordiniert werden, sodass der Anwender sich ausschließlich um die Regeldefinition und um die abzubildende Umgebung durch Geräte, Zustände und weitere potentielle Mittel innerhalb eines smarten Büros kümmern muss.

1.4 Forschungsstrategie und Forschungsmethoden

Dieser Abschnitt der Arbeit widmet sich der Forschungsstrategien und der darauf angewendeten Forschungsmethoden. Hierbei werden die Strategien kurz erläutert und die Methoden skizziert. Die Anwendung der Strategien findet zu späterem Zeitpunkt statt.

Die Struktur ist in vier Phasen Forschung, Konzeption, prototypische Umsetzung und Evaluation aufgeteilt. Zuerst erfolgt durch ein systematisches Literaturreview eine Analyse zum aktuellen Stand der Technik in Bezug auf die Forschungsfrage. Schwerpunkt dabei liegt auf der einfachen Handhabung der formalisierten Interaktionen eines Softwareentwicklers bei der Umsetzung, bzw. Ergänzung einer bestehenden Softwarelösung zur Abdeckung weiterer Anwendungsfälle. In der Phase der Anforderungserhebung werden Experteninterviews durchgeführt, mit denen die Anforderungen an das Produkt, für welches ein Konzept erstellt wird, identifiziert werden. Ergänzend dazu werden im Rahmen der Arbeit zusätzliche Anforderungen durch das Requirements Engineering (RE) und der Anwendung des *user-centered design*-Prinzips² sowie des *target group analysis*-Ansatzes (4.2) ermittelt. Dabei wird die Nutzerorientierung auf den Softwareentwickler ausgelegt. Basierend auf den Ergebnissen wird das Konzept erstellt und daraufhin die Umsetzung eingeleitet. Zur anschließenden Evaluation wird ein Usability-Test und ein erneutes Experteninterview durchgeführt, damit Eindrücke über das Konzept entstehen und die Erfahrungen der Experten während des Usability-Tests in die

²Iterativer Prozess zur Ermittlung von Anforderungen, die nutzerorientiert aufgestellt werden. <https://www.interaction-design.org/literature/topics/user-centered-design> Abgerufen am 08.05.2022

Beantwortung der Forschungsfrage mit einfließen.

Die soeben genannten Forschungsmethoden werden nachfolgend näher erklärt.

1.4.1 Experteninterview

Zu Anfang wird dem Experten der Hintergrund des Interviews erläutert und ihm zum Sachverhalt relevante Fragen gestellt.

Die Interviewfragen können als offene oder geschlossene Fragen formuliert werden, eine beliebige Reihenfolge an Antworten ist möglich oder nur eine begrenzte Anzahl [ROBSON 2002]. Der Verlauf des Interviews kann von dem Forscher selbst bestimmt werden. Sind nur bestimmte Antworten erwünscht, so können die Fragen strukturiert und geschlossen formuliert werden.

Ebenso gibt es den semi-strukturierten und den unstrukturierten Ansatz, bei dem das Interview offen gestaltet werden kann. Der semi-strukturierte Ansatz eignet sich, um spezielle Fragen zu adressieren, bzw. eine Reihenfolge festgelegt ist. Mit dem unstrukturierten Vorgehen wird ein völlig offenes Interview angestrebt, bei dem der Verlauf abhängig vom Inhalt der Konversation ist.

Die Durchführung von Experteninterviews zählt zu den qualitativen Forschungsansätzen zum Sammeln von Daten in bestimmten Kontexten.

1.4.2 Systematisches Literaturreview

Mit einem systematischen Literaturreview wird zur evidenzbasierten Identifizierung, Bewertung und Interpretation von bestehender Literatur eine wissenschaftliche Methode angewendet, mit der Fragestellungen zu einer bestimmten Thematik herausgearbeitet werden können. Mithilfe dieses Ansatzes soll die Erzielung einer Schlussfolgerung zu einem untersuchten Objekt ermöglicht werden. Die Methodik des systematischen Literaturreviews orientiert sich an den Richtlinien von Kitchenham et al.. Diese ist in drei aufeinander aufbauende Schritte gegliedert. Zu Beginn erfolgt die Planung, anschließend die Durchführung und abschließend die Dokumentation und Offenlegung der Ergebnisse [KITCHENHAM und CHARTERS 2007].

1.4.3 Usability-Test

Ein Usability-Test ist eine Methode zur Testung von Prototypen oder Arbeitsversionen von Computerschnittstellen [LAZAR, FENG und HOCHHEISER 2017]. Das Testen der Nutzbarkeit kann durch vorab definierte Aufgaben, die durch Probanden und Benutzer in einer dafür vorgesehenen Umgebung durchgeführt werden, erfolgen. Diese können je nach Anwendungsfeld oder -kontext unterschiedlich ausfallen. Ebenso können solche Tests auch dazu beitragen, physische Interaktionen mit bspw. Geräten zu bewerten.

Alle Ansätze für Usability-Tests haben ein grundlegendes Ziel: Die Qualität einer Schnittstelle zu verbessern und sie anwenderfreundlich zu gestalten [LAZAR, FENG und HOCHHEISER 2017]. Während diese Tests optimalerweise Schnittstellenfehler aufdecken, die den Benutzern Schwierigkeiten bereiten können, ist gleichzeitig eine Modifizierung des Schnittstellendesigns für zukünftige Entwicklungen.

Zur Kategorisierung von durchgeführten Tests, werden Werkzeuge verwendet, die das Verhalten und die Meinungen der Benutzer messbar machen. Eines davon ist das *System Usability Scale (SUS)*³. Dabei handelt es sich um einen einfachen technologieunabhängigen Fragebogen, anhand dessen die Gebrauchstauglichkeit eines Systems bewertet werden kann. Dieser ist eine Praktik zur quantitativen Analyse der Nutzbarkeit. Der Umfang umfasst zehn Fragen nach der Likert-Skala [LIKERT 1932].

1.5 Aufbau der Arbeit

Die vorliegende Master-Thesis gliedert sich nach den soeben genannten einleitenden Information im Aufbau in insgesamt zehn Kapitel. Das erste Kapitel (1) beschreibt die Motivation (1.1) zur Bearbeitung dieser Thematik rundum IoT und Smart Home. Darauffolgend wird die Forschungsfrage (1.2) im Rahmen der Thesis definiert. Nach der Beschreibung der Forschungsfrage wird im anknüpfenden Abschnitt (1.3) die Zielsetzung der Arbeit erläutert. Hierbei werden zusätzliche Schwerpunkte aufgegriffen.

Das Kapitel (2) widmet sich den essentiellen und wichtigen Grundlagen dieser Arbeit. Zu Anfang wird dem Leser der Terminus des Internet of Things (2.1) zum besseren Verständnis näher gebracht, gefolgt von einer Einführung in die Thematik des Smart Home (2.2), der historischen und kontinuierlichen Entwicklung und den Zielen, die mit der Verwendung verfolgt werden sollen. Mit dem Verständnis der übergeordneten Begriffe, IoT und Smart Home, werden Technologien (2.3) aufgegriffen, die im Rahmen dieser Arbeit erwähnenswert sind und verwendet werden. Abschließend werden im Kapitel (2) die Softwarelösungen, Home Assistant und openHAB (2.4 & 2.5), dargestellt. Diese dienen als Grundlage für die Evaluation und werden im Kapitel (7) der im Rahmen dieser Arbeit konzipierten Lösung gegenübergestellt.

Die theoretischen und methodischen Hintergründe sowie der Stand der Technik wird in Kapitel (3) beleuchtet. Dieser Teil enthält Beschreibungen, Forschungen und aktuelle Erkenntnisse über Technologien, die im Umfeld der Smart Home Anwendungen innerhalb des IoT eingesetzt werden.

Kapitel (4) befasst sich mit den Anforderungen, engl. Requirements, an das zu entwickelnde System, die aus der Markt- und Zielgruppenanalyse, den Anwendungsfällen und den Experteninterviews generiert werden.

Nach Aufbereitung der Anforderungen durch das sogenannte Anforderungsmanagement, engl. *Requirements Engineering*, wird in Kapitel (5) das Konzept erarbeitet, welches als Grundlage für die prototypische Implementierung dient.

In Kapitel (6) wird die Umsetzung des Konzepts skizziert, Herausforderungen aufgegriffen und Lösungen erarbeitet. Es wird aus praktischer Sicht die Architektur und deren Komponenten betrachtet.

³System Usability Scale. <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html> Besucht am 03.07.2022

Das erzielte Ergebnis und Resultat wird abschließend zusammengefasst.

Die Evaluation (7) behandelt die objektive Betrachtung und Bewertung des entstandenen Prototyps anhand von Usability-Tests und damit zusammenhängenden Experteninterviews. Zudem wird die Forschungsfrage evaluiert und beantwortet.

Im vorletzten Teil, Kapitel (8), wird ein Fazit aus den Erkenntnissen und Ergebnissen gezogen. Dieses Schlussresümee führt nochmals die Höhepunkte sowie eine eigene Einschätzung auf.

Zum Abschluss der Thesis wird in Kapitel (9) ein Ausblick aufgezeigt, der Aufschluss über die Erweiterungsmöglichkeiten und der Innovation der in dieser Thesis erfolgten Arbeit gibt.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für diese Thesis relevanten Grundlagen geschaffen, um ein Grundverständnis und fundiertes Wissen über verwendete Technologien zu erlangen und die nachfolgende Recherche, Konzeption und Umsetzung besser verstehen zu können.

2.1 Internet der Dinge

Das Internet der Dinge (IdD), im Englischen Internet of Things (IoT), zählt als eines der Schlagworte in der Informationstechnologie (IT). In der Domäne des IoT bekommen Gegenstände und Objekte eine eindeutige Identität, die eine Kommunikation miteinander als auch das Entgegennehmen von Befehlen erlaubt. Mit dem Internet der Dinge lassen sich Anwendungen sowie Prozesse automatisieren und Aufgaben erledigen ohne das von außen Eingegriffen werden muss [LUBER und LITZEL 2016]. Die Prozessautomatisierung findet sich auch im Kontext des Smart Home wieder, welches in nachfolgendem Kapitel genauer aufgegriffen wird.

In der einschlägigen Literatur gibt es für das Internet of Things keine allgemeingültige Definition, die alle Anwendungsbereiche abdeckt. Die Definitionen und Auslegungen der Interpretation unterscheiden sich je nach Anwendungsgebiet. Demnach gibt es viele verschiedene Forschungsgruppen, darunter Forscher, Akademiker, Innovatoren, Entwickler und Geschäftsleute, die den Begriff oder die zugrundeliegende Problemstellung definiert haben. Die Ursprünge jedoch sind dem Experten für digitale Innovationen, Kevin Ashton¹, zuzuschreiben.

Die in der Literatur auffindbaren Definitionen verfolgen zwei Sichtweisen. Zum einen die aktive Sicht, d. h. die Daten sind von Menschen erstellt, zum anderen die passive, bei der die Daten von Dingen, darunter die Sensoren und Aktoren, erstellt werden [MADAKAM, RAMASWAMY und TRIPATHI 2015]. Eine aus dem Zusammenhang hervorgehende, aus dem wissenschaftlichen Artikel entnommene Definition ist folgende:

¹ Britischer Technologie-Pionier, Mitgründer des Auto-ID Centers am Massachusetts Institute of Technology (MIT). https://de.wikipedia.org/wiki/Kevin_Ashton (Abgerufen am 22.03.2022)

“An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment” [MADAKAM, RAMASWAMY und TRIPATHI 2015]

Daraus kann die Ableitung erfolgen, dass der Begriff des Internet der Dinge für die Vernetzung von Gegenständen im privaten Gebrauch sowie von industriellen Geräten und Maschinen über das Internet steht. Damit Geräte individuell angesprochen werden können, werden diese mit einer eindeutigen Identität, genauer einer Internetprotokoll (IP)-Adresse, im Netzwerk belegt und mit elektronischer Intelligenz ausgestattet [LUBER und LITZEL 2016]. Darüber können die Netzwerkteilnehmer über das Internet kommunizieren und Prozesse automatisiert erledigen. Die sogenannten *intelligenten Geräte* werden auch oft mit dem englischen Begriff, *Smart Devices*, betitelt.

Neben der Kommunikation der Geräte untereinander kann ebenso entweder über das Gerät selbst oder eine zentrale Schnittstelle via Internet interagiert werden. Dadurch sind Objekte und Gegenstände durch einen Benutzer von beliebigen Orten aus auch außerhalb des Netzwerks erreich- und bedienbar. Diese Art und Weise wird auch in dem zentralen Thema des Smart Home verwendet. Die Funktion als auch die Umsetzung wird im Kapitel (2.2) näher beleuchtet.

Das Internet der Dinge ist ein elementarer Baustein der IT-Welt. Mit dem IoT wird die Vision verfolgt, eine globale Infrastruktur zu erstellen, mit der physische Objekte miteinander vernetzt werden und jeder Zeit zur Verfügung stehen. Das Internet of Things kann auch als globales Netzwerk angesehen werden, indem die Kommunikation zwischen Mensch zu Mensch, Gerät zu Mensch und Gerät zu Gerät ermöglicht wird. Viele Forschungsartikel sprechen von der Verschmelzung der digitalen und physischen Welt.² Die Vereinigung beider Welten ist die Verknüpfung physischer Objekte, die eindeutig identifizierbar sind, mit einer virtuellen Repräsentation in einer vergleichbaren Internet-Struktur.

Gesamtbild des Internet of Things

Der folgenden Abbildung (2.1) ist zu entnehmen, welche Technologien rund um das Internet der Dinge liegen und in Verbindung damit stehen.

²Das Internet der Dinge – der digitale Zwilling der Welt. Kompetenzzentrum Öffentliche IT in Kooperation mit dem Fraunhofer Institut. <https://www.oeffentliche-it.de/trendsonar-iot> Abgerufen am 23.03.2022.

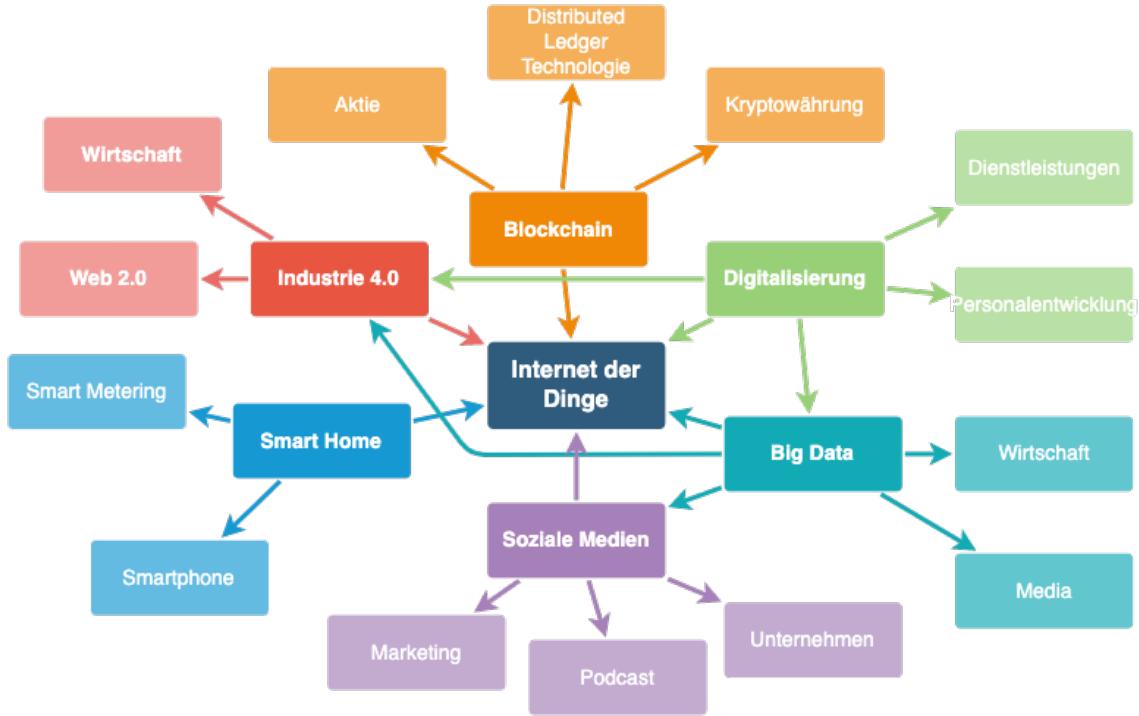


Abbildung 2.1: Technologische Einordnung von IoT [SIEPERMANN und LACKES 2018]

Beispielsweise ist das IoT eine wesentliche Grundlage für das Themengebiet *Big Data*. Die durch Sensoren und Akteure erzeugten Daten können eine Grundlage für die Verwendung im Bereich *Big Data* sein. Dabei werden die Datenmengen gespeichert und mithilfe von Mustern und Herangehensweisen des Big Data³ analysiert. Big Data ist kein Bestandteil dieser Arbeit und wird demnach nicht weiter ausgeführt. Das Beispiel dient lediglich zu Veranschaulichung und Interpretation der oben aufgeführten Abbildung (2.1).

Eine allgemeine exemplarische Skizzierung eines Systems, welches nach dem IoT-Konzept aufgebaut ist, kann der folgenden Abbildung (2.2) entnommen werden:

³Definition und Funktionsweise von Big Data. <https://www.oracle.com/big-data/what-is-big-data/> Abgerufen am 25.03.2022

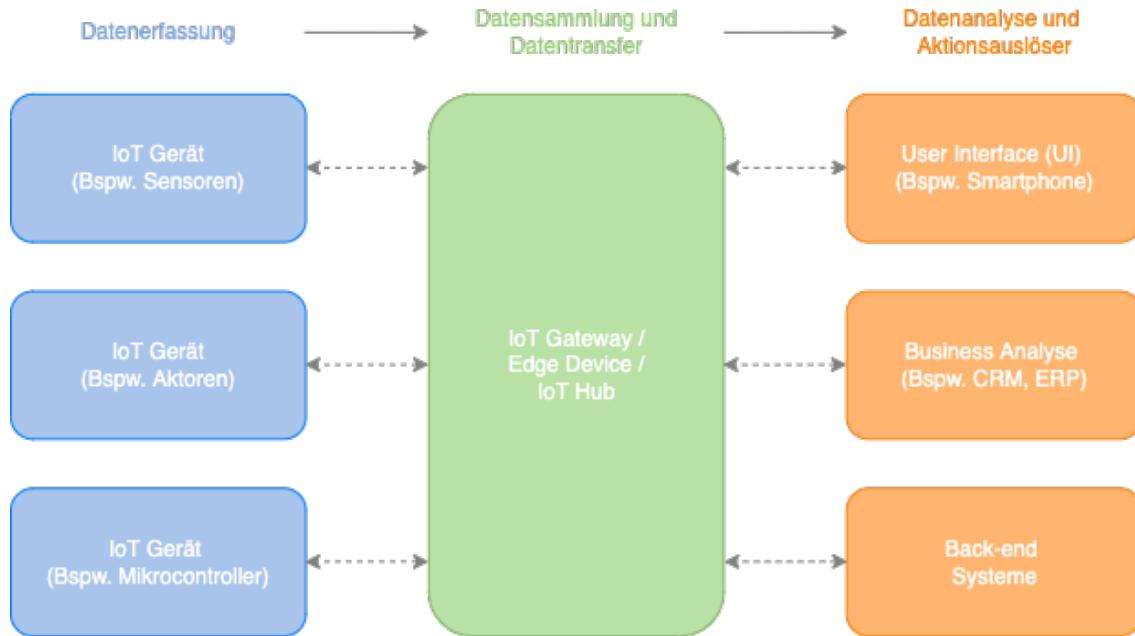


Abbildung 2.2: Exemplarische Darstellung eines IoT-Systems [GILLIS 2022]

Hierbei werden die jeweiligen Komponenten verdeutlicht, die in einem System zu finden sind. Mit den *IoT-Geräten*, darunter beispielsweise Sensoren und Aktoren, findet die Datenerzeugung statt. Mit dem dahinterstehenden *Gateway* werden die aus den Geräten erzeugten Daten gesammelt und an zentraler Stelle an die Cloud gesendet. Nach der Datensammlung können diese an verschiedene Komponenten zur weiteren Verarbeitung gesendet werden, die durch die Visualisierung über das Smartphone stattfinden kann oder zur Durchführung von Prozessanalysen dient. Ebenso können die Daten an weitere Backend-Systeme zur Weiterverarbeitung übermittelt werden.

Anwendungsbereiche des IoT

Grundlegend kann im Bereich des Internet of Things zwischen zwei Anwendungsbereichen unterschieden werden. Dies ist zum einen der private Bereich und zum anderen der industriellen. Der private Bereich deckt hauptsächlich die Thematik rund um den Gebrauch von Alltagsgegenständen ab und deren Vernetzung untereinander zur komfortableren und intelligenteren Nutzung der Geräte. Darin inbegriffen sind Gebäudeautomatisierungen und Ereignissesteuerungen über das Internet. Diese Funktionen sind Hauptbestandteil des Smart Home-Konzeptes, welches in Abschnitt (1.5) genauer aufgegriffen wird.

Der industrielle Bereich beschäftigt sich mit der Vernetzung von Maschinen und Anlagen miteinander, so dass sich ganze industrielle Prozesse automatisieren lassen und so die Effizienz der Prozess- und Produktionsabläufe gesteigert werden. Die Nutzung des IoT im industriellen Bezug ist

ein elementarer Bestandteil der heutigen *Industrie 4.0*⁴, auch bekannt als *Industrial Internet of Things, dt. Industrielles Internet der Dinge (IIoT)*.

Im Rahmen dieser Arbeit wird zwischen den beiden Anwendungsbereichen differenziert und den Fokus auf den privaten Bereich gelegt.

Mit dem Internet der Dinge-Ansatz gibt es zwei Paradigmen, die in Kombination als auch alleinstehend Anwendung finden. Diese werden im folgenden Abschnitt kurz erläutert.

Edge und Cloud Computing

Bei den beiden Paradigmen handelt es sich zum einen um Edge Computing und zum anderen um Cloud Computing. Ziel beider Ansätze ist das Verwalten von Daten, bzw. das Arbeiten mit erzeugten Daten.

Das Edge Computing verfolgt einen *dezentralen Ansatz*, bei dem die Berechnung und Erhebung der Daten direkt auf dem Gerät durchgeführt wird. Dies bedeutet, dass jedes Gerät über eine eigene Intelligenz verfügt, bei der Daten direkt nach der Erzeugung verarbeitet und gespeichert werden können. Zu einem späteren Zeitpunkt können in Form einer Datenbündelung oder einer Vorauswahl die Informationen an ein Rechenzentrum weitergegeben werden.

“Edge computing is different from traditional cloud computing. It is a new computing paradigm that performs computing at the edge of the network. Its core idea is to make computing closer to the source of the data [...]” [CAO u. a. 2020]

Bei Cloud Computing handelt es sich grundlegend um Bereitstellungen von Computerdiensten, Kapazitäten, Ressourcen und der Rechenleistung über das Internet, die nicht in einem eigenen Rechenzentrum verwaltet und betrieben werden müssen. Die Verwaltung wird dem Anbieter des Cloud Computing überlassen. Hauptsächlich steht dabei die flexible Ressourcennutzung, Skalierung und Verteilung von Rechenleistungen im Vordergrund. In Zusammenhang mit den verfügbaren Ressourcen werden auch verschiedene Modelle, Cloud Typen und diverse Dienste angeboten⁵. Diese werden ihm Rahmen der Arbeit nicht weiter ausgeführt.

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” [MELL und GRANCE 2011]

Mit dem Cloud Computing wird im Vergleich zum Edge Computing ein zentraler Ansatz verfolgt, bei dem die erzeugten Daten von den Akteuren und Sensoren direkt an eine zentrale Stelle gelangen. Von dort aus findet die Verarbeitung und Analyse der Daten statt.

Eine Kombination beider Ansätze vereint deren Vorteile und ist als *Hybrid Cloud* bekannt. Diese Konstellation ist in den meisten Architekturen wiederzufinden.

⁴Definition und Beschreibung der Industrie 4.0 <https://www.plattform-i40.de/IP/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html> Abgerufen am 25.03.2022

⁵Einblick in die Cloud-Umgebung von dem Anbieter Microsoft. <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/> Abgerufen am 28.03.2022

2.1.1 Paradigmen und Kommunikationsmodelle

Damit eine umfassende Grundlage im Bereich IoT geschaffen wird, behandelt folgender Abschnitt Paradigmen und Kommunikationsmodelle, die in dieser Umgebung verwendet werden. Für die Darstellung dieser Modelle wird eine Literaturquelle genutzt, die ein paar der gängigsten und prägnantesten Architekturen und Modelle abgedeckt. Die Repräsentation der Interaktionsparadigmen sind den Schaubildern (2.3) zu entnehmen.

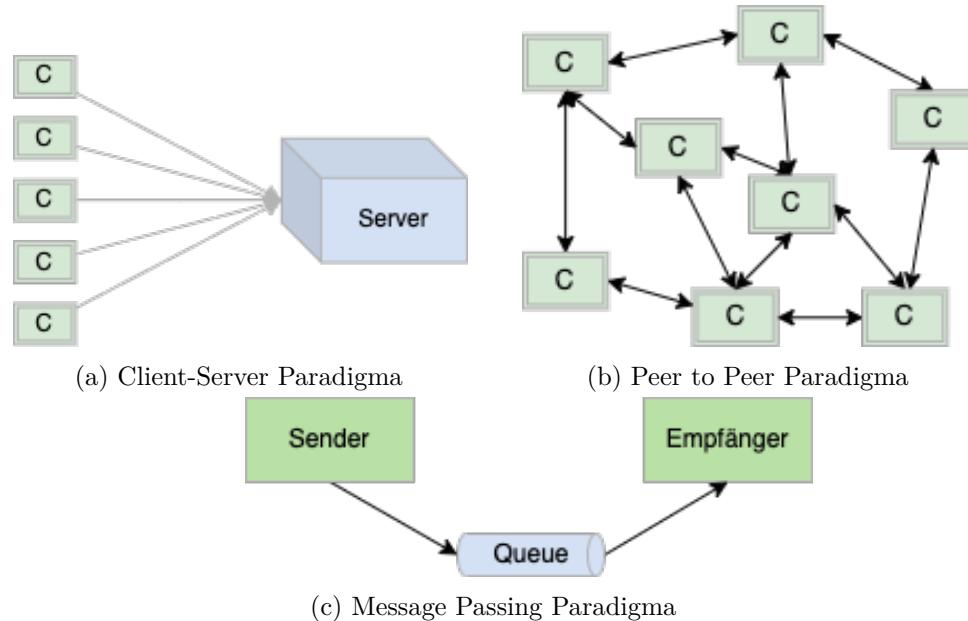


Abbildung 2.3: Client-Server, Peer-to-Peer und Message Passing Interaktionsparadigma [MINERVA, BIRU und ROTONDI 2015]

Name	Beschreibung
Client-Server (2.3a)	Basierend auf einer sehr einfachen Interaktion zwischen Clients und dem Server. Ein Client sendet eine Anfrage an den Server und erwartet dementsprechend eine Antwort.
Peer to Peer (P2P) (2.3b)	Gegensatz zu Client-Server Konzept. Hierbei können Geräte sowohl als Client Dienste abfragen, als auch als Server Dienste anbieten.
Message Passing (2.3c)	Basierend auf einer einfachen Organisation in der ein Absender eine Nachricht mittels einer Warteschlange (Queue) an einen Empfänger weiterleitet.

Tabelle 2.1: Interaktionsparadigmen nach [MINERVA, BIRU und ROTONDI 2015]

Eine Analyse und detailliertere Darstellung der jeweiligen Ansätze kann der Ausarbeitung, aus der die Paradigmen verwendet wurden, entnommen werden.

Im Zusammenhang mit Architekturen, die in dem Bereich IoT Anwendung finden, und Interaktionsparadigmen wird häufig zur Unterstützung verschiedener Interaktionsmodelle auf Protokolle gesetzt, die Daten und Primitiven transportieren. Diese Kommunikationsmodelle sind in zwei Bereiche, die ihre Anwendung in Machine to Machine (M2M) finden, kategorisiert:

- Simple Object Access Protocol (SOAP)⁶: Dabei handelt es sich um ein Standardprotokoll, das dafür entwickelt wurde, um verschiedene Programmiersprachen auf verschiedenen Plattformen untereinander kommunizieren zu lassen.
- Representational State Transfer (REST)⁷: Dabei handelt es sich um eine Reihe von Architekturprinzipien, die auf die Anforderungen leichtgewichtiger Webdienste und mobiler Anwendungen abgestimmt sind. Anfragen, die über diese Form gesendet werden, können in mehreren Formaten beantwortet werden, darunter beispielsweise Hypertext Markup Language (HTML), Extensible Markup Language (XML) und JavaScript Object Notation (JSON).

Beide verfolgen das Ziel der Datenübermittlung zwischen Web-Anwendungen über eine Schnittstelle, das sogenannte Application Programming Interface (API). REST Architekturen werden bei einem IoT-Szenario aufgrund ihrer Simplizität und ihres Komforts bevorzugt. [MINERVA, BIRU und ROTONDI 2015]

2.1.2 Historische Entwicklung

Die ersten Konzepte zu dem heute bekannten IoT liegen schon einige Jahrzehnte zurück. Die historische Definition des Internet of Things wurde im Jahr 1999 von Kevin Ashton geprägt. Von der Definition bis hin zum erstmaligen Einsatz, bzw. zur Umsetzung eines Konzepts vergingen noch ein paar Jahre. Ein kurzer Einblick in die chronologische Entstehung des IoT ist der folgende Tabelle zu entnehmen:

⁶Detailliertere Definition des Begriffs SOAP. <https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest> Abgerufen am 29.03.2022

⁷Detailliertere Definition des Begriffs REST. <https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest> Abgerufen am 29.03.2022

Jahr	Industrielle Beteiligung und Zuordnung
1970	Der erste Vorschlag zu miteinander verbundenen Geräten und Maschinen
1990	John Romkey und Simon Hackett entwickelten den ersten Toaster, der über das Internet an- und ausgeschaltet werden konnte. ⁸
1995	Das Unternehmen Siemens leitete das erste Mobilfunkmodul ein, welches für die Kommunikation zwischen Maschinen zuständig ist (M2M-Technologie).
1999	Die Definition des Begriffs IoT von Kevin Ashton wurde veröffentlicht und von der Gesamtheit akzeptiert, als er bei Procter & Gamble (P&G) an den sogenannten Radio-Frequency Identifitaction (RFID) Chips arbeitete. RFID-Chips verwenden elektromagnetische Felder, die an Objekte angebracht sind, um diese identifizieren und verfolgen zu können. Ein solches System besteht aus einem Funktransponder, einem Funkempfänger und -sender.
2004 - 2005	Der Begriff wurde in angesehenen Publikationen verwendet, darunter dessen von Boston Globe und The Guardian. International Telecommunication Union (ITU) veröffentlichte deren ersten Bericht über das Thema.
2008 - 2011	Die Definitionen und Publikationen fanden erste Anwendungen in praktischen Umsetzungen. Gartner Inc. nahm den Begriff in ihre Recherche-Arbeiten mit auf.

Tabelle 2.2: Historische Entwicklung vom Internet der Dinge [DURGA, PROF und KUMAR 2020]

Mittlerweile ist das Internet der Dinge ein fester Bestandteil der Industrie, ebenso im privaten Umfeld. Viele Szenarien und Konzepte werden erarbeitet und umgesetzt. Die Kommunikation von Maschinen untereinander ist in der heutigen Zeit fester Bestandteil der Industrie und wird immer weiter ausgebaut. Forschergruppen sind daran interessiert, mehrere Gebiete und Anwendungsbereiche innerhalb des IoT zu erschließen. Stark vertreten ist dabei das Fraunhofer Institut für Integrierte Schaltungen (IIS)

2.1.3 Ziele von IoT

Die Definition von dem Internet der Dinge gibt die Richtung an, in die sich die Zielsetzung von IoT bewegt. Davon ist die Intention abzuleiten, dass die Lücke zwischen der realen und der virtuellen Informationswelt so gering wie möglich gehalten, bzw. zunehmend minimiert wird. Dieser Informationsbedarf oder auch die Informationslücke besteht, da in der realen Welt Objekte, Dinge und Gegenstände einen bestimmten Zustand vorzeigen. Ein Beispiel dazu ist „die Temperatur liegt bei 35 Grad Celsius“ oder „das Licht ist an und hat 75 % Helligkeit“. Diese Informationen sind in der virtuellen Welt nicht ohne weiteres verfügbar. Das Bestreben demnach, welches auch zum Großteil im Bereich Smart Home abgedeckt wird, ist, dass viele reale Gegenstände ihre Zustandsinformationen für die Weiterverarbeitung in der virtuellen Welt, beziehungsweise im Netzwerk zur Verfügung stellen. Diese Informationen können beispielsweise Aussagen zu der aktuellen Nutzung, zur bestimmten Positionierung im Raum und über die Alterung eines Gerätes geben. Diese Zustandsinformationen können dazu beitragen, dass die Nutzung des Gegenstandes vom Anwender ausgewertet und so

optimiert werden kann, z.B. durch die Erkennung eines Defekts oder einer Automatisierung, damit die Ressource effizient genutzt wird.

2.2 Smart Home

Smart Home, im Deutschen „*intelligentes Zuhause*“, ist eine Domäne des IoT. Diese Rubrik der Anwendung widmet sich überwiegend dem Gebrauch von sämtlichen Haushaltsgeräten und -einrichtungen. Ein kleiner Ausschnitt solcher Nutzgegenstände sind unter anderem Lampen, Kontaktoren, Thermostate, Service-Roboter, Staubsauger-Roboter, Kühlschränke Geräte rundum die Haussicherheit.

Unter dem Oberbegriff Smart Home ist eine Weise zu verstehen, mit der die Erhöhung der Wohn- und Lebensqualität erzielt, Energieverbrauch unter Verwendung vernetzter und fern steuerbarer Geräte effizienter gestaltet, Sicherheit gesteigert und Abläufe verschiedener Prozessschritte automatisiert werden kann.

Der Begriff *intelligentes Zuhause* wird verwendet, wenn die Haustechnik und Haushaltsgeräte untereinander vernetzt sind. Die Definition im Deutschen Gebrauch, welche nach (Strese et al. 2010) in der Untersuchung im Rahmen der wissenschaftlichen Begleitung zum Programm „*Next Generation Media (NGM) des Bundesministeriums für Wirtschaft und Technologie*“ aufgegriffen wird, lautet wie folgt:

„Das Smart Home ist ein privat genutztes Heim (z.B. Eigenheim, Mietwohnung), in dem die zahlreichen Geräte der Hausautomation (wie Heizung, Beleuchtung, Belüftung), Haushaltstechnik (wie z.B. Kühlschrank, Waschmaschine), Konsumelektronik und Kommunikationseinrichtungen zu intelligenten Gegenständen werden, die sich an den Bedürfnissen der Bewohner orientieren. Durch Vernetzung dieser Gegenstände untereinander können neue Assistenzfunktionen und Dienste zum Nutzen des Bewohners bereitgestellt werden und einen Mehrwert generieren, der über den einzelnen Nutzen der im Haus vorhandenen Anwendungen hinausgeht.“ [STRESE u. a. 2010]

Eine vergleichbare Definition wurde zu späterem Zeitpunkt durch eine Literaturrecherche publiziert. Diese beschreibt die zugrundeliegende Thematik weniger aus Anwendersicht sondern widmet sich vielmehr dem System und der Konnektivität.

„A smart home is a place with heterogeneous systems to many front devices with the support of embedded information and communication architectures[...]“ [BALAKRISHNAN, VASUDAVAN und MURUGESAN 2018]

Den beiden Definitionen ist zu entnehmen, dass die Kernaussage eine ähnliche ist, es jedoch in Büchern, Fachartikeln, Publikationen an Universitäten und in den verbreiteten Medien bis heute keine durchgängige gibt. Aus der einschlägigen Literatur wird ersichtlich, dass viele Synonyme für die Benennung der Thematik verwendet werden, darunter beispielsweise [STRESE u. a. 2010]:

- Connected Home
- Elektronisches Haus
- Intelligentes Haus (engl. Smart House)
- Smart Living
- Home of the Future

Eine elementare Information im Zusammenhang dieser Arbeit ist die Verwendung des Begriffs *intelligentes Büro*, die ebenso in den Kontext des Smart Home gehört. Hierbei wird lediglich die Räumlichkeit im unternehmerischen Jargon verwendet, die ebenso eine Grundlage für die Verwendung von Komponenten des Smart Home bietet. Im Abschnitt (2.2.1) wird nochmals konkret darauf eingegangen.

Es wird deutlich, dass die Verwendung des Begriffs als auch die zugrundeliegenden technischen Verfahren weiträumig einsetzbar sind und deshalb die Begriffsdefinition nicht eindeutig festgehalten werden kann.

Teilsysteme des Smart Home

Der primäre Anwendungsbereich des Smart Home ist die Automatisierung häuslicher Prozesse. Dadurch sollen dem Nutzer in vielerlei Hinsicht Aufwände erspart und Informationen zentralisiert angezeigt werden. Die Hausautomatisierung umfasst eine Menge von Teilsystemen. Ein Ausschnitt dieser Teilsysteme ist der folgenden tabellarischen Auflistung zu entnehmen:

Segment	Beschreibung
Licht	Beleuchtung, Lichtmanagement/Szenarien, Storen/Rollos
Zutritt	Zutrittskontrolle, Klingelanlage, Schlosser, Anwesenheits- und Bewegungserfassung
Überwachung	Technische Alarme: Feuer, Rauch, Gas; Intrusion: Glasbruchmelder, Video; Babyphon, Urlaubswachschutz
Notfall	Sprinkleranlage, unabhängige Stromversorgung, Fluchtwegsystem
Metering	Verbrauchszähler für Strom, Gas, Wasser, Wärme, uvm.
Konsumelektronik	TV, Internet, Smartphones, Tablets, Spielekonsolen etc.
Hausgeräte	Kühlschrank, Waschmaschine, Staubsauger, Service-Roboter; Hausgerätemonitoring, -diagnostik, und -fernbedienung
Heimlogistik	Einkaufs- und Speiseplanung, häusliche Dienste
Hobby	Haustierversorgung, Aquarienmanagement, etc.
Mobilität	PKW mit Diagnistik, Navigationssystem mit local based services, Info-/Entertainmentangebote etc.

Tabelle 2.3: Teilsysteme des Smart Home [STRESE u. a. 2010]

Die Ausstattung des Smart Home mit Intelligenz und die Vernetzung der Teilsysteme untereinander haben zum Ziel, die Aufgaben möglichst autonom mit wenig übergeordneter, zentraler Steuerung abzuarbeiten. Dabei müssen die erzeugten als auch erforderlichen Daten mit anderen Komponenten des Gesamtsystems ausgetauscht werden.

Eine mögliche Vernetzung und auch Verwendung solcher Komponenten wird in folgender Abbildung (2.4) skizziert. Diese Grafik dient als grobe Übersicht potentieller Anwendungsszenarien, repräsentiert jedoch nicht alle Möglichkeiten der Anwendung. Darüber hinaus gibt es weitaus mehr

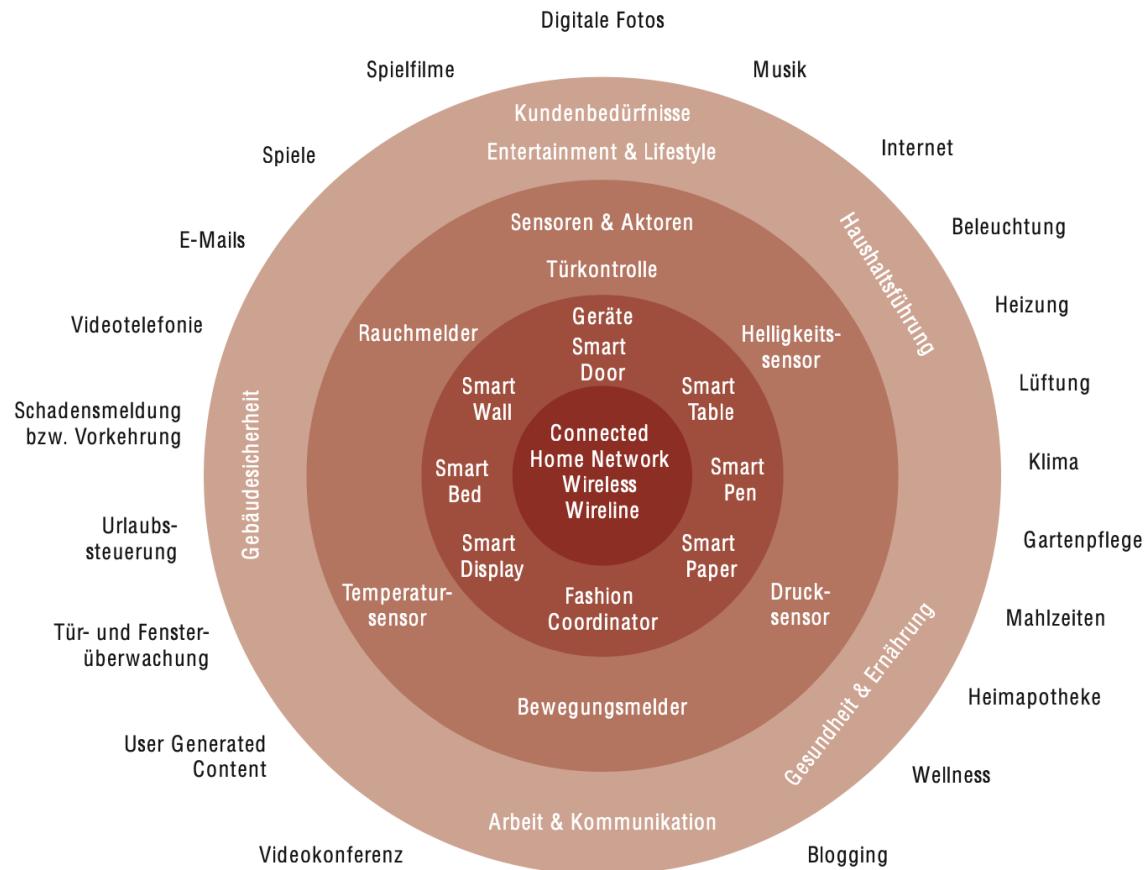


Abbildung 2.4: Mögliche Anwendungsszenarien im Smart Home [STRESE u. a. 2010]

Anwendungsszenarien, beziehungsweise werden diese in der Abbildung (2.4) in einem Überbegriff zusammengefasst. Ein immer bedeutenderes Anwendungsszenario ist die Kopplung von Robotern jeglicher Art, darunter die schon weit verbreiteten Staubsaugerroboter oder vor allem Service-Roboter, die immer mehr in die Thematik des Smart Home zu integriert versucht werden.

Einordnung von Smart Home in das Internet der Dinge

Im Konsumentenmarkt wird die Technologie des IoT in Produkten eingesetzt, die das Konzept des Smart Home verfolgen. Diese beinhalten Haushaltsgeräte und -ausstattung, wie beispielsweise Thermostate, Sensoren, Sicherheitssysteme und Lampen, die mehrere Systeme und Übertragungstechnologien unterstützen. Dazu zählen unter anderem Plattformen, darunter Google Nest, Apple HomeKit und Amazon Alexa uvm., ebenso die Übertragungstechnologien, die im Abschnitt (2.3) aufgegriffen werden. Weitere Beispiele sind der Tabelle (2.3) zu entnehmen.

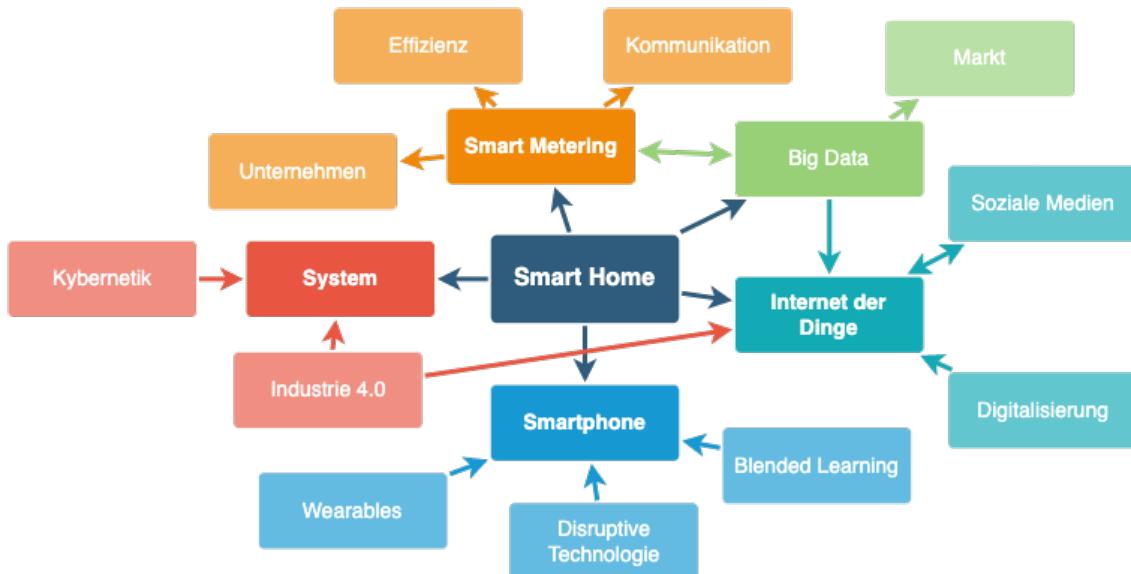


Abbildung 2.5: Technologische Einordnung von Smart Home in Verbindung zu IoT [BENDEL 2021]

Die Abbildung (2.5) zeigt die Verbindungen, als auch die Beziehungen von Smart Home zu anderen Technologien. Hier wird deutlich, dass im Bereich des Smart Home viele Fragestellungen thematisiert werden können, die andere Themenbereiche tangieren. Mit inbegriffen ist zum Beispiel die Komponente Smartphone, da dieses genutzt wird, um als Fernbedienung zu fungieren und Prozesse und Automationen anzeigen und überwachen zu können. Der Bereich des Smart Metering⁹ deckt die Messung von Verbrauchsdaten ab. Dabei handelt es sich um intelligente Messsysteme, die Daten zum Verbrauch, darunter Strom, Gas und Wasser, erheben und diese von den jeweiligen Anbietern zur Rechnungsstellung genutzt werden. Ein Beispiel dazu ist ein digitaler intelligenter Stromzähler mit direkter Kommunikationsmöglichkeit zum Anbieter selbst.

Der Aufbau eines Smart Home ist architektonisch ähnlich zu dem Grundprinzip einer IoT-Lösung. Die Veranschaulichung des zugrundeliegenden Aufbaus ist der Abbildung (2.2) zu entnehmen. Um die Bezugspunkte zu IoT und die Einordnung zu untermauern, wird in folgendem Abschnitt auf die Funktionsweise von Smart Home eingegangen.

⁹Smart Metering ist das computergestützte Messen, Ermitteln und Steuern von Energieverbrauch und -zufuhr. <https://wirtschaftslexikon.gabler.de/definition/smart-metering-53998> Abgerufen am 06.04.2022

Funktionsweise eines Smart Home

Ein Smart Home System besteht aus mehreren Teilsystemen, die in der Regel aus verschiedenen Komponenten bestehen. Wichtige Elemente eines grundlegenden Aufbaus sind die Endgeräte, die sogenannten Aktoren, Eingabegeräte, Sensoren, Gateway und die Vernetzung über Funk, Kabel oder Stromnetz. Die Endgeräte sind die Ausgabegeräte, die über die intelligente Steuerung ansprochen werden können. Darunter zählen zum Beispiel LED-Lampen, Rolläden, Lüftungsanlagen, Lautsprecher, Fernseher, Waschmaschinen und jegliche Arten von Service-Robotern. Eingabegeräte sind die Schnittstelle zwischen der Interaktion des Nutzers und des Smart Home Systems. Das können Wandschalter, Touchdisplays, Fernbedienungen, Smartphones und Regler sein. Mithilfe dieser Schnittstelle können Zustände und Aktionen an den Endgeräten ausgelöst werden. Bei einer fehlenden Verbindung zwischen den Steuerelementen sind diese trotzdem noch über direkte Schaltbefehle steuerbar. Damit die Zustände ebenso digitalisiert werden können und dem System zur Verfügung stehen, werden Sensoren benötigt. Diese greifen die physikalischen oder elektronischen Eigenschaften des Endgerätes ab, um die Zustände zu ermitteln. Das Gateway repräsentiert die zentrale Steuereinheit, auf dem die Sensordaten eingehen und die Sendung von Befehlen an die Aktoren stattfindet. Ebenso ermöglicht das Gateway die Kommunikation der Endgeräte und Sensoren untereinander. Eine mögliche Internetverbindung zwischen dem Gateway und einer zentralen Plattform, die über die Cloud erreichbar ist, kann ebenso hergestellt werden. Je nach Gerät kann über das Gateway auch eine direkte Steuerung einzelner Elemente stattfinden. Das letzte Element, die Vernetzung, ist für die Verbindung aller Elemente zuständig. Hierfür kommen verschiedene Protokolle, die unter anderem in Abschnitt (2.3) beschrieben werden, per Funk, Kabel oder Stromnetz zum Einsatz.

Ein exemplarischer Aufbau der Komponenten ist der folgenden Abbildung zu entnehmen:

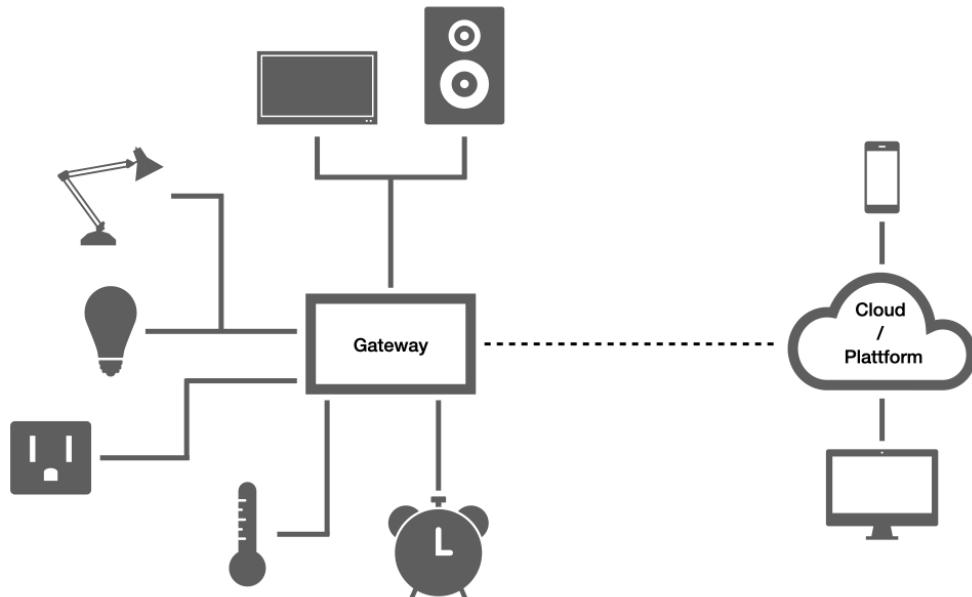


Abbildung 2.6: Aufbau und Funktionsweise einer Smart Home Infrastruktur

2.2.1 Smart Office - Intelligentes Büro

Ein Smart Office, im Deutschen intelligentes Büro, ist ein konkreter Anwendungsbereich des Smart Home. Während das Smart Home vorwiegend von der Verbesserung der Lebensqualität, der Einsparung von Energiekosten, der Optimierung des Wohlbefinden im privaten Wohnraum und der Erleichterung des Alltags des Nutzers handelt, zielt das Smart Office darauf ab, das Wohlbefinden und die Produktivität der Mitarbeiter zu steigern. Weitere wichtige Punkte sind neben den beiden aufgeführten auch die Erhöhung der Sicherheit der Büroräume und der Nachhaltigkeit durch z.B. Reduzierung des Stromverbrauchs. In diesem Zusammenhang spielt die Automation von Gebäudemanagement- und die damit verbundenen Sicherheitssysteme eine große Rolle.

Viele Funktionalitäten und Vorteile sind deckungsgleich zu denen des Smart Home. Durch intelligente Geräte, Aktoren und Sensoren ist die Steuerung von Licht, Raumtemperatur etc. möglich. Hervorzuheben ist unter anderem die mögliche Zutrittskontrolle per Gesichtserkennungssysteme, die an zentraler Stelle mit weiteren Komponenten kommunizieren und beliebige zusätzliche Schritte einleiten können. Ebenso zu nennen ist der Aspekt in Richtung Gebäudesicherheit und deren Ausbau, denkbar in Form von intelligenten Feuer- und Wassermeldern oder weiteren Maßnahmen, die durch Automationen abgebildet werden können.

In unserer heutigen Zeit ist die Nachhaltigkeit und Ressourceneffizienz von immer größerer Bedeutung. Automatische Abschaltungen von Beleuchtungen, Kaffeemaschinen, Monitoren oder anderen Geräten durch Präsenz- und Bewegungsmelder oder Remotesteuerungsoptionen können Ressourcen- und Energieeinsparungen umsetzen.

2.2.2 Historische Entwicklung

Im Bereich des Smart Home wurde im Jahr 1975 die erste Netzwerktechnologie für Hausautomationen präsentiert und vermarktet. Bekannt wurde diese unter dem Namen *xX0*¹⁰. Dabei handelt es sich um ein stromleistungsbasiertes Netzwerkprotokoll zur Gebäudeautomation. Die Schaltsignale werden über die Hausinstallation, das Stromnetz des Hauses, transportiert. Eingeführt wurde die Technologie von dem Unternehmen Busch-Jaeger unter dem Namen *Timac X10*. Es zeichnete sich durch die einfache Konfiguration und dessen interessanten Funktionen zu diesem Zeitpunkt aus [ASCENDORF 2014]. Die Weiterentwicklung des Systems, Timac X10, fand im Jahre 1998 statt, indem von Busch-Jaeger ein neues Produkt Namens *Powernet EIB* in Deutschland eingeführt wurde. Dieses baute ebenso auf dem grundlegenden Netzwerkprotokoll X10 auf und fügte sich nahtlos in den europäischen Installationsbus (EIB/KNX) ein [BUSCH-JAEGER 2021]. KNX ist ein Bussystem zur Gebäudeautomation, welches basierend auf dem EIB weiterentwickelt wurde. Es zählt heute noch zu den kabelgebundenen Standards. Im Jahre 2001 eröffnete die Fraunhofer IMS in Kooperation mit der Universität Duisburg-Essen die Fraunhofer-inHaus-Forschungsanlage [FRAUNHOFER 2021]. Innerhalb dieser Institution erforschten, entwickelten, testeten und demonstrierten Dienstleister, Hersteller und Nutzer mit dem Fraunhofer-Institut und der Universität neue Systemlösungen und weitere Produktkomponenten sämtlicher Arten im Bereich des Wohnens. Anfang des Jahres 2005 wurde die deutsche Telekom in dem Smart Home Bereich aktiv und präsentierte der Öffentlichkeit

¹⁰[https://de.wikipedia.org/wiki/X10_\(Protokoll\)](https://de.wikipedia.org/wiki/X10_(Protokoll)) - X10 Protokoll Erklärung. Abgerufen am 06.04.2022

ein vollständig vernetztes, „intelligentes“ Musterhaus in Berlin, das sogenannte *T-Com-Haus*. Anfang des Jahres 2012 wurde das Bundesministerium für Wirtschaft und Klimaschutz (BMWK) im Sektor des Smart Home aktiv. Seitdem fördert die Behörde das „*Zertifizierungsprogramm Smart Home + Building*“, bei dem Forschungen im Bereich des Smart Home von Vertretern akademischer Einrichtungen und Industrieunternehmen durchgeführt werden. Ziele dabei sind unter anderem die Erstellung von Standards und Prüfsiegeln für systemübergreifende Interoperabilität der Geräte eines Smart Home [V. 2013]. 2013 präsentierte die deutsche Telekom neue Lösungen in einem weiteren Musterhaus in Darmstadt. Darin finden sich mehrere Komponenten und Vernetzungen als im Berliner Modell [BRAJKOVIC 2014], so die Nutzung verschiedener Funkstandards, die es ermöglichen, intelligente Geräte unterschiedlichster Hersteller zu konfigurieren und über ein Smartphone, Tablet oder Computer zu kontrollieren und zu steuern [BRAJKOVIC 2014]. Die Auswirkungen der rasanten Entwicklung dieser Technologie trieben das Interesse in die Höhe, sodass 2014 und 2015 die Technologie das Hauptthema der Internationale Funkausstellung (IFA) bei vielen Ausstellern war. Bis heute wird in dem Segment des Smart Home geforscht und immer weitere Funktionalitäten entwickelt. Der Trend der Nutzung von intelligenten Geräten ist weiter zunehmend. Die Marktsituation und der aktuelle Ist-Stand wird in dem Abschnitt Marktanalyse (4.1) nochmals aufgegriffen und mit Statistiken und Umfragen belegt.

2.2.3 Ziele von Smart Home

Die Ziele der intelligenten Vernetzung sind in erster Linie diejenigen, die die großen Domänen abdecken [LUBER und LITZEL 2019]:

- Komfort (Steuerung, Fernbedienung, Delegation und Automationen)
- Energie- und Kosteneffizienz
- Sicherheit (Überwachung)

Allgemein lässt sich der Komfort durch die entfernungsunabhängige, bequeme Steuerung von Licht, Heizung, Unterhaltungselektronik, Service-Robotern und vielen weiteren Geräten steigern. Hierfür spielen beispielsweise Smartphones, zentrale Steuerungsgeräte und Tablets eine wichtige Rolle. Diese fungieren ähnlich wie eine herkömmliche Fernbedienung. Es können ebenso Prozesse automatisiert oder durch ein bestimmtes vorgegebenes Ereignis, welches eintreten kann, angestoßen werden. Mit diesen Automationen können gewünschte Wohnbedingungen zu bestimmten Anlässen und Zeiten erzeugt werden. Beispielsweise kann die Heizung vor Eintreffen reguliert werden, um die optimale Temperatur zu erreichen. Die Steuerung von Geräten spielt gleichermaßen eine Rolle in den Bereichen der Energie- und Kosteneffizienz und der Sicherheit.

Durch das Verbinden mehrerer Komponenten mithilfe einer Automation können Energiekosten verringert werden, wenn bspw. ein Temperaturabfall durch ein offenes Fenster von Sensoren gemessen wird und diese ein Signal übermitteln. Dementsprechend kann die Heizung für den Zeitraum ausgeschaltet oder reguliert werden, um überflüssige Wärmeerzeugung zu vermeiden. Weitere Einsparungen können durch das Ausschalten von aktuell nicht benötigten Ressourcentypen, wie z.B. Elektrogeräten oder LED-Lampen, vorgenommen werden.

Zuletzt ist die Erhöhung der Sicherheit und das Überwachen von Eingängen und Fenstern zu nennen. Die Bewegungsmelder und Sensoren können Aktivitäten registrieren und sind Auslöser für die Weitergabe der Information an den Eigentümer. Zusätzlich können mit Kameras Objekte überwacht werden, beispielsweise ein Haustier, das kurze Zeit alleine zuhause ist oder es können Einbruchversuche registriert und gemeldet werden. Auch eine Alarmierung bei Flutung des Kellers oder bei Hausbrand sind denkbar.

Die Geräte im Smart Home erreichen weitestgehend diese Ziele und werden stetig weiterentwickelt, verbessert und um neue Funktionalitäten ergänzt.

2.3 Technologien

Die bereits angesprochene Kommunikation und Vernetzung zwischen Geräten basiert im Allgemeinen auf diversen Protokollen. Um diese Datenbewegung und Kommunikation besser verstehen zu können, werden im Folgenden bekannte Protokolle erwähnt und aufgeführt und eines der meist verwendeten näher betrachtet. Um einen Vergleich herzustellen, wird ein vergleichbares Protokoll betrachtet. Diese werden dann zum Abschluss gegenübergestellt.

2.3.1 Übertragungsmethoden

Allgemein gibt es im Bereich des Smart Home mehrere Methoden und Möglichkeiten, die Objekte miteinander zu vernetzen. Dazu gehören Protokolle über Bluetooth, Ethernet, WLAN, Bussysteme, Funk und Stromleitung. Jeder Hersteller setzt diese abhängig von seinem Produkt passend ein. Proprietäre Systeme funktionieren nur über eine Übertragungsmethode. So erzwingen die Hersteller die Nutzung einer Produktlinie, bzw. den Kauf von Elementen eines einheitlichen Systems. Geräte, die die Möglichkeiten besitzen, über mehrere Protokolle zu kommunizieren, sind flexibler einsetzbar und mit mehreren Plattformen und Geräten kompatibel. Grundlegend werden mit diesen Übertragungsmethoden Netzwerke erstellt, über das die Geräte in einem Smart Home kommunizieren können.

Die Auflistung der zum aktuellen Zeitpunkt am meist verwendeten Übertragungsmethoden dient lediglich als Einblick in die verschiedensten Ausprägungen der Kommunikationstechnologie. Demnach wird im Rahmen dieser Arbeit das Thema nicht ausführlich vertieft. Ein Ausschnitt der populärsten Übertragungsmethoden wird in folgender Tabelle aufgelistet¹¹.

¹¹ Auswahl von derzeit verwendeten Übertragungsmethoden. https://de.wikipedia.org/wiki/Smart_Home#Übertragungsmethoden Abgerufen am 02.04.2022

Technologie	Übertragung	Frequenzbereich (Funk)	Proprietär
ZigBee	Funk	2,4 GHz, 868 MHz	Nein
Z-Wave	Funk	868 MHz	Nein
HomeMatic	Funk / Datenleitung	868,3 MHz	Ja
KNX	Funk / Strom- und Datenleitung	868 MHz / -	Nein / Ja (Datenleitung als Gesamtsystem)
Wi-Fi / WLAN	Funk	2,4 - 5 GHz	Nein
Bluetooth	Funk	2,4 GHz	Nein
io-homecontrol	Funk	868-870 MHz	Ja

Tabelle 2.4: Übertragungsmethoden des Smart Home

2.3.2 MQTT

Das Message Queue Telemetry Transport (MQTT)-Protokoll ist eines der ältesten offenen Netzwerk- und Nachrichtenprotokolle der M2M-Kommunikation. Dies wurde 1999 von IBM Mitarbeiter Andy Stanford-Clark¹² und von Cirrus Link Solutions Mitarbeiter Arlen Nipper¹³ entwickelt. Die Technologie ermöglicht die Übertragung von Messdaten, sogenannten Telemetriedaten, in Form von Nachrichten zwischen Maschinen und Geräten. Die erzeugten Messdaten durch beispielsweise Sensoren und Aktoren können durch ihre minimale Größe und die kompakte Form des Protokolls in einem kleinen Datenpaket auch bei hoher Verzögerung oder bei beschränktem Netzwerk übertragen werden [NAIK 2017]. MQTT ist ein klassisches Client-Server-Protokoll, welches nach dem *Publish-/Subscribe* Kommunikationsmodell entwickelt wurde. Ein MQTT-Client veröffentlicht Nachrichten an einen MQTT-Server, den sogenannten MQTT-Broker. Diese können von anderen Clients abonniert oder auf dem Broker für zukünftige Abonnements aufbewahrt werden. Jede erzeugte Nachricht wird an eine Adresse veröffentlicht, die als Thema, im Englischen *Topic*, bezeichnet wird [NAIK 2017]. MQTT-Clients können mehrere Topics abonnieren und erhalten jede Nachricht, die an das jeweilig abonnierte Topic gesendet wird.

Die Leichtgewichtigkeit des Protokolls ermöglicht es, die Nachrichten bei eingeschränkter Netzwerkverfügbarkeit zu übermitteln. Ausschlaggebend dafür ist das binär-basierte Protokoll, welches normalerweise einen festen Header von zwei Bytes mit kleinen Nachrichtennutzlasten von maximal bis zu einer Größe von 256 MB [NAIK 2017] enthält. Grundlegend ist MQTT auf der Basis des Transportprotokoll Transmission Control Protocol (TCP) aufgebaut und nutzt zur Verstärkung der Sicherheit die Transport Layer Security (TLS)/Secure Sockets Layer (SSL)-Verschlüsselung. Dadurch sind Client und Broker mit ihrer Kommunikation verbindungsorientiert. Die Anwendung

¹²Informationen zu Herrn Stanford-Clark <https://stanford-clark.com> Abgerufen am 12.04.2022

¹³Informationen zu Herrn Nipper <https://www.inductiveautomation.com/resources/podcast/the-co-inventor-of-mqtt-arlen-nipper-from-cirrus-link-solutions> Abgerufen am 12.04.2022

von MQTT im Bereich des Smart Home zeichnet sich durch die Möglichkeit aus, große Netzwerke mit vielen kleineren Geräten, die von einem Backend-Server, bzw. dem Backend-System überwacht und gesteuert werden müssen, zu betreiben. Dennoch ist es nicht für Multicast-Daten oder Übertragungen von Gerät zu Gerät ausgelegt. Die Nutzbarkeit von MQTT ist aufgrund der wenigen Steueroptionen und der Einfachheit des Messaging-Protokolls sehr simpel und leichtgewichtig [NAIK 2017].

Ein interessanter Aspekt des MQTT-Brokers ist, dass er die gesamte Datenlage seiner Kommunikationspartner aufbewahrt und so die Option bietet, zusätzlich als Zustandsdatenbank betrieben werden zu können. Weniger leistungsfähige Geräte können dadurch mit einem MQTT-Broker verbunden werden und Befehle und Daten entgegennehmen, zugleich entsteht aber ein komplexeres Lagebild auf dem Broker. So können Daten an einen leistungsfähigeren Kommunikationspartner weitergeleitet und dort ausgewertet werden.¹⁴

Auf diese Weise können durch das Message Queue Telemetry Transport-Protokoll Automatisierungs-lösungen geschaffen werden, wodurch im Segment IoT und Smart Home eine einfache Verwendung ermöglicht wird. Dementsprechend fand bis heute eine schnelle Verbreitung statt.

Zusammengefasst ist der MQTT-Broker die Kommunikationsschnittstelle der Smart Home-Geräte und der Smart Home-Plattform. Alle Kommunikationspartner können so Informationen (Nachrichten / Messages) auf bestimmte Topics (Endpunkten) senden und diese abonnieren (publish / subscribe).

Publish/Subscribe Kommunikationmodell

Das Prinzip des Publish/Subscribe Kommunikationsmodells besteht darin, dass Komponenten, die an vorgegebenen Topics interessiert sind, bestimmte Informationen konsumieren und ihr Interesse anmelden [HUNKELER, TRUONG und STANFORD-CLARK 2008]. Dieser Vorgang wird als Abonnement (subscription) bezeichnet. Geräte, die an dem Vorgang oder an bestimmten Informationen interessiert sind, werden als Abonnenten (subscriber) definiert. Im Gegenzug können Geräte und Komponenten bestimmte Informationen produzieren und diese veröffentlichen (publish) und über den Markler (broker) an bestimmte Abonnenten weitergeben. Diese Vermittlung der Informationen zwischen Herausgeber (publisher) und Abonnent (subscriber) erfolgt über den Markler, welcher sämtliche Abonnements koordiniert. Alle Abonnenten müssen sich explizit bei dem Broker anmelden, um die Informationen zu erhalten [HUNKELER, TRUONG und STANFORD-CLARK 2008].

¹⁴<https://mqtt.org> Abgerufen am 13.04.2022



Abbildung 2.7: Themenbasiertes Pub/Sub Kommunikationsmodell [HUNKELER, TRUONG und STANFORD-CLARK 2008]

Dieses Prinzip ist die Grundlage des MQTT-Protokolls. Im Folgenden wird ein Beispiel aufgezeigt, welches die Kommunikation über das Publish/Subscribe Modell darstellt.

Beispiel

Damit am Beispiel der Steuerzentrale, die im Rahmen dieser Arbeit konzipiert und prototypisch implementiert wird, ein Prozess gestartet werden kann, müssen bestimmte Ereignisse durch MQTT-Nachrichten eintreten. Das hier verwendete Beispiel ist das Öffnen einer Büroeingangstür. Hierbei wird von einem Relais an der Tür, welches das Türschloss steuert, eine Nachricht an den *Broker* herausgegeben:

`1 publish -topic: Buero/Tuer/Zustand -message: "offen"`

Code-Beispiel 2.1: Erzeugung und Veröffentlichung einer Nachricht

Diese Nachricht wird von der Steuerzentrale konsumiert und weiterverarbeitet. Damit der Informationskanal (topic) über die Steuerzentrale verfügbar ist, muss dieser Kanal konsumiert werden:

`1 subscribe -topic: Buero/Tuer/Zustand`

Code-Beispiel 2.2: Empfang und Konsum einer Nachricht

Mit der empfangenen Nachricht kann dann über die Steuerzentrale ein Prozess, welcher vorab als Regel implementiert wurde, ausgelöst werden, beispielsweise eine Durchsage im Büro.

2.3.3 AMQP

Das Advanced Message Queue Protocol (AMQP)-Protokoll ist ebenso ein leichtgewichtiges M2M-Protokoll, welches im Jahre 2003 von John O'Hara JPMorgan Chase in London, Großbritannien, entwickelt wurde [NAIK 2017]. Der Fokus dieses Protokolls liegt auf der Unternehmens-Messaging-Ebene. Es wird hohen Wert auf die Zuverlässigkeit, Sicherheit, Bereitstellung und Interoperabilität der Kommunikation gelegt. AMQP unterstützt neben der Publish/Subscribe- auch die Request/Response Architektur. Es bietet eine Bandbreite an Funktionen im Zusammenhang mit Messaging, wie z. B. Queuing, themenbasiertes Publish-and-Subscribe-Messaging, flexibles Routing und Transaktionen [NAIK 2017]. Das Kommunikationsmodell nach dem AMQP Standard erfordert, dass der Herausgeber (publisher) oder der Empfänger (subscriber) einen *Austausch (exchange)* mit einem bestimmten Namen generiert und diesen dann sendet [NAIK 2017]. Die beiden Komponenten, Empfänger und Herausgeber, nutzen den Namen zum Austausch und zum Verbindungsauftbau. Der Empfänger erstellt darauf eine Warteschlange (queue) und hängt diese an den Austausch an. Nachrichten, die über diese Verbindung ausgetauscht werden, müssen über einen gesonderten Prozess (binding) mit der Warteschlange abgeglichen werden [NAIK 2017].

Das binäre Protokoll AMQP erfordert einen Header von acht Byte mit Nachrichtennutzlasten. Die Größe der Nachricht ist abhängig von dem Broker, bzw. dem Server. Die verbindungsorientierte Kommunikation von AMQP baut auf dem Standard-Transportprotokoll TCP auf. Durch das TLS/SSL-Protokoll wird die Kommunikationssicherheit gewährleistet. Ein Wesensmerkmal des AMQP Kommunikationsmodells ist die Zuverlässigkeit [NAIK 2017].

Der direkte Vergleich der beiden Protokolle, MQTT und AMQP, ist dem Anhang (A) beigefügt.

Neben den beiden aufgeführten Kommunikationsmodellen gibt es noch weitere, darunter das klassische Hypertext Transfer Protocol (HTTP) und das Constrained Application Protocol (CoAP), die allerdings im Rahmen dieser Arbeit nicht weiter ausgeführt werden.

2.4 Home Assistant

Eine der etablierten Smart Home Plattformen ist das sogenannte Home Assistant System. Die Open-Source-Software ist ein zentrales Steuerungssystem von Heimautomationen, ebenso ist eine Verwaltung von intelligenten Geräten mit dem Fokus der lokalen Steuerung und Sicherung der Privatsphäre möglich. Der Zugriff kann über die Smartphone-App, jeweils verfügbar für iOS und Android, oder auch über die webbasierte Benutzeroberfläche (Web-App) erfolgen. In dem lokalen System können auch Geräte per Sprachbefehl gesteuert werden. Kompatible Plattformen sind unter anderem Google Assistant, Amazon Alexa und Apple HomeKit. Home Assistant bietet eine vielfältigere Verknüpfung von Geräten, Services und Plattformen. Die zentrale Steuerung unterstützt durch modulare Integrationskomponenten die einzelnen Geräte, Anwendungen und Services. Für die drahtlose Kommunikation werden native Integrationskomponenten verwendet, darunter Bluetooth, ZigBee und Z-Wave. Diese werden verwendet, um lokale Personal Area Network (PAN) mit Geräten mit geringem Stromverbrauch aufzubauen. Die Steuerung kann auch mit proprietären Ökosystemen stattfinden, sofern diese eine offene API oder Anbindungen über MQTT anbieten.¹⁵ Die Platform ist in Python geschrieben, wird aktiv instand gehalten und durch eine große Community unterstützt. Die Software ist allgemein unter der Apache 2.0, veröffentlicht. Der folgende Abschnitt befasst sich in Kürze mit der Historie des Systems.

Historie

Anfang des vierten Quartals im Jahr 2013 startete das Python-Projekt von Paulus Schoutsen und wurde im November 2013 erstmals auf GitHub veröffentlicht.¹⁶

Vier Jahre nach den ersten Entwicklungen der Smart Home Plattform wurde im Juli 2017 ein verwaltetes Betriebssystem mit dem Namen *Hass.io* implementiert.¹⁷ Dadurch gelang der Durchbruch der vereinfachten Verwendung der Home Assistant Plattform auf kleineren Computern, sogenannten Einplatinencomputern, wie beispielsweise ein Gerät der Raspberry Pi Serie. In Zusammenhang mit dem Betriebssystem kam ein *Supervisor*-Verwaltungssystem hinzu, das den Benutzern die Verwaltung, Sicherung und Aktualisierung der lokalen Installation erleichtert. Ein weiteres Feature des Supervisors ist die Möglichkeit, der Plattform über Add Ons weitere Funktionalitäten zur Verfügung zu stellen.¹⁸

Die Software wird stetig weiterentwickelt und verbessert. Mittlerweile gehört sie zu den am meisten genutzten Open-Source-Plattformen im Bereich Smart Home.

¹⁵Definition von Home Assistant. https://en.wikipedia.org/wiki/Home_Assistant Besucht am 16.04.2022

¹⁶Anfänge von Home Assistant. <https://www.linux.com/topic/embedded-iot/home-assistant-python-approach-home-automation/> Besucht am 18.04.2022

¹⁷Verkündigungen von Home Assistant. <https://www.home-assistant.io/blog/categories/announcements/> Besucht am 18.04.2022

¹⁸Einstieg in das Hass.io Betriebssystem. <https://www.home-assistant.io/blog/2017/07/25/introducing-hassio/> Besucht am 18.04.2022

2.4.1 Konzept und Architektur

Home Assistant bietet eine Plattform für die zentrale Haussteuerung und die damit einhergehende Steuerung von Heimautomationen. Die Software ist nicht nur eine einfache Steuerungs- und Konfigurations-Software, sondern ein eingebettetes Betriebssystem, das verbraucher- und nutzerorientiert das Verwenden und Konfigurieren von Haussteuerungen erleichtert.¹⁹

Damit der offene Ansatz von Home Assistant gegenüber Anbietern von intelligenten Geräten nicht eingeschränkt ist, bietet die Software Möglichkeiten, um viele Geräte zu integrieren und über die Plattform zu nutzen. Somit begegnet Home Assistant der Heterogenität des offenen Marktes, insofern, dass diese Geräte auf gemeinsame Konzepte gebracht werden. Diese sind in vier Konzepte aufgeteilt, mit der die Vereinheitlichung vorangetrieben werden kann:

- Integration (Integration): Integrationen repräsentieren die Geräte und Dienste innerhalb der Home Assistant Anwendung. Ebenso können mittels Integrationen auch Daten von Datenpunkten abgerufen werden.
- Gerät (Device): Nach der Konfiguration der Integration werden die Geräte in Home Assistant angelegt. Diese werden dann als erkannte Geräte der Integration dargestellt, z.B. als Temperatur-, Licht- oder Feuchtigkeitssensor.
- Entität (Entity): Die Datenpunkte sind die Geräte, die sogenannten Entitäten, die durch die Integrationen standardisiert werden. Dies sind Objekte, die Funktionalitäten oder Daten des Gerätes darstellen, z.B. die Temperatur, Helligkeit oder die Feuchtigkeit.
- Automatisierung (Automation): Automatisierungen sind Prozesse, die bei einem bestimmten ausgelösten Event ausgeführt werden sollen. Dieses Auslöser (trigger) können Zeitpunkte, Ereignisse oder manuell gesteuerte Aktionen des Nutzers sein, z.B. das Ausschalten des Bürolichts, wenn durch einen Bewegungssensor fünf Minuten keine Bewegung festgestellt wurde oder wenn der Helligkeitswert, der über den Sensor festgestellt wurde einen bestimmten Wert erreicht hat, soll das Licht ebenso ausgeschaltet werden.

Die Architektur der Home Assistant Anwendung ist grundlegend als eingebettetes System eines Betriebssystem aufgestellt, welches in drei Schichten aufgeteilt ist. In unterster Ebene befindet sich das Betriebssystem, welches als minimales Linux System postiert ist, um die darauf liegenden Schichten, den Aufseher (supervisor) und den Kern (core), zu betreiben. Mit dem Supervisor wird das Betriebssystem verwaltet und konfiguriert. Der eigentliche Kern interagiert mit dem Supervisor, den Geräten und den Services.

Der Supervisor ist die Schicht über dem Betriebssystem. Die Kommunikation der beiden Komponenten findet über einen D-Bus statt. Diese Zwischenschicht ermöglicht dem Benutzer die Verwaltung der Home Assistant Installation.

Die Aufgaben des Supervisors sind wie folgt definiert²⁰:

¹⁹Konzept und Architektur von Home Assistant. https://developers.home-assistant.io/docs/architecture_index Besucht am 19.04.2022

²⁰Architektur des Supervisors. <https://developers.home-assistant.io/docs/supervisor> Besucht am 22.04.2022

- Dieser führt den Home Assistant Kern (Core) aus.
- Dieser führt die Updates des Home Assistant Core aus.
- Dieser führt einen *Rollback* bei fehlgeschlagenem Update durch.
- Dieser führt Sicherungen und Wiederherstellungen durch.
- Dieser verwaltet die Add Ons der Core Instanz.

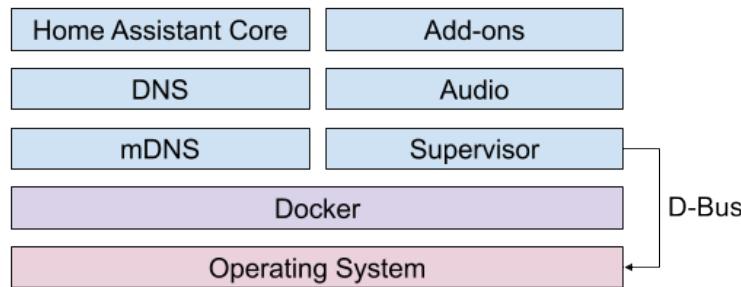


Abbildung 2.8: Architektur des Home Assistant Supervisors [SCHOUTSEN 2020]

Die auf dem Supervisor aufbauende Architektur ist die der Anwendung, der sogenannte Core. Dieser besteht aus vier Komponenten, welche den Hauptteil abbilden: Event Bus, State Machine, Service Registry und Timer²¹.

Mit dem Event Bus wird das Abhören und Auslösen von Events und Ereignisse erleichtert. Die Komponente stellt eine Eigenschaft der Home Assistant Anwendung dar. Die Zustandsmaschine (State Machine) ist eine weitere Komponente, mit der die Zustände der Gegenstände, wie Sensoren, Service- oder Saugroboter uvm., überwacht und deren Änderungsereignisse an den Event Bus ausgelöst werden. Mit der Dienstregistrierung (Service Registry) wird der Event Bus auf eingehende Aufrufe von Diensten abgehört. Über die Service Registry kann der Benutzer Dienste hinzufügen und verwalten. Mittels der Entwicklertools, die über das User Interface (UI) aufgerufen werden können, kann der Benutzer die Dienste und Automatisierungen anwählen und konfigurieren. Das letzte Element, der Timer, ist ebenso eine Komponente der Architektur, die zeitveränderte Ereignisse gemäß einer gegebenen Frequenz an den Event Bus sendet. Somit können zeitbasierte Automatisierungen vereinfacht und ausgelöst werden. Als Datenbank wird eine nicht cloudbasierte SQLite²² Datenbank verwendet. Diese ist nur auf dem Gerät enthalten und wird nicht über das Internet übertragen. Im lokalen Netzwerk hat der Nutzer die Möglichkeit, auf die Datenbank zuzugreifen und eine Historie von Aktionen einzusehen. Eine Verlaufskomponente, die ebenso im Core enthalten ist, speichert die Ereignisse innerhalb der Plattform. Somit können Nutzer auf alle gespeicherten Informationen zu Hause zugreifen und diese einsehen [LI u. a. 2018].

²¹Entwickler Dokumentation der Plattform. <https://developers.home-assistant.io/docs/> Besucht am 24.04.2022

²²Structured Query Language, eine Datenbanksprache zur Definition, Abfrage und Bearbeitung von Datenstrukturen in relationalen Datenbanken

Home Assistant Core Architecture

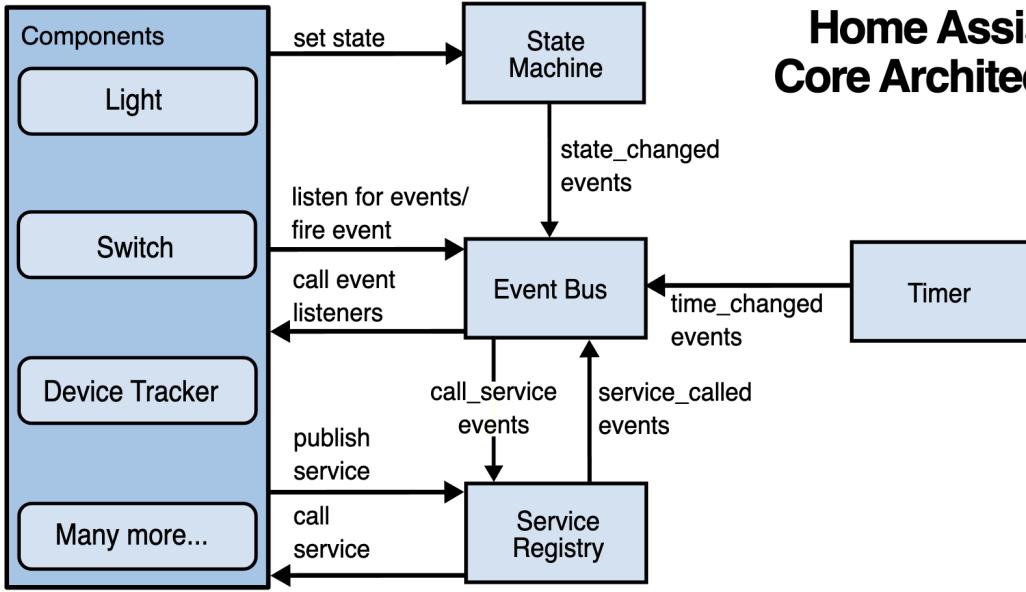


Abbildung 2.9: Architektur des Home Assistant Cores [LI u. a. 2018]

2.4.2 Ziele und Schwerpunkte

Jede Softwarelösung verfolgt bestimmte Ziele und Schwerpunkte, um die Zielerreichung, die sog. Definition of Done (DoD), messbar zu machen. Die mit der Home Assistant Plattform verfolgten Ziele sind unter anderem der Schutz der Privatsphäre und die Datensicherheit, die Wahlmöglichkeit zur Anbindung von Geräten und die Haltbarkeit der Plattform²³.

Der Schutz der Privatsphäre wird durch Home Assistant gewährleistet, indem das System selbst nur im eigenen Netzwerk betrieben wird und es Geräteanbietern somit nicht ermöglicht wird, Daten abzugreifen und daraus Verhaltensmuster zu analysieren. Home Assistant setzt voraus, dass die Möglichkeit der Datenschnittstelle über das Internet optional eingestellt werden kann.

Home Assistant bietet den Vorteil einer breitgefächerten Auswahlmöglichkeit von Geräten und Verknüpfungen, sodass der Endnutzer zwischen verschiedenen Geräteherstellern wählen kann, z.B. Bewegungssensoren von Ikea, LED-Leuchten von Philips und Thermostate von Netatmo und diese wahllos über Automatisierungen kombinieren kann.

Mit den von Home Assistant verfolgten Zielen gehen folgende Schwerpunkte einher:

Dazu gehört die Kontrollierbarkeit über eine einzige zentrale Stelle und die Konfiguration von

²³Schwerpunkte der Home Assistant Plattform. <https://www.home-assistant.io/blog/2021/12/23/the-open-home/> Besucht am 24.04.2022

Automationen, die Prozesse selbstständig auslöst. Durch die Möglichkeit der Automatisierung und Steuerung von Komponenten im Eigenheim, bzw. im Büro kann der Komfort erhöht als auch die Energie- und Stromkosten gesenkt werden. Die Kompatibilität der Plattform ermöglicht die Verwendung von mehreren Geräten und Herstellern über eine zentrale Stelle, der Home Assistant Software. Dies ermöglicht dem Nutzer die uneingeschränkte Verwendung von Geräten und die Unabhängigkeit zu Herstellern, die ggf. die Geräte preislich über dem Marktdurchschnitt anbieten. Ein weiterer Schwerpunkt ist die Reduzierung des Datenkommunikationsflusses über das Internet, die durch das geschlossenen System mit wenig Kommunikationsschnittstellen nach außen erreicht werden kann. Ein Indiz dafür ist die zugrundeliegende lokale Datenhaltung über eine Datenbank, die direkt auf der Hardware läuft, auf der ebenso die Plattform betrieben wird.

2.4.3 Stärken und Schwächen

Die Stärken und Schwächen der Plattform belaufen sich auf die folgenden Aspekte:

Stärken	Schwächen
Open-Source-Software	Schwerer Einstieg für neue Nutzer im Hinblick auf die Konfiguration der Automationen
Privatsphäre & Datenschutz	Automationen im YAML-Datenformat und benutzerdefinierte Komponenten können für neue Benutzer schwierig sein
Große Community	Langwierige Installationen von Integrationen
Automationen sind sehr mächtig	Die Erweiterung, Einrichtung und Konfiguration erfordert die Bearbeitung von YAML-Dateien und einen Neustart bei jeder Änderung
Stetige Aktualisierung, Weiterentwicklung und Fehlerbehebung	Automationen können kompliziert aufgebaut sein
Unterstützung vieler Geräte (Herstellerunabhängigkeit) und großflächige Kompatibilität	Nutzung von ZigBee und Z-Wave erfordert jeweils eigene Konfigurationen, welche die All-in-One Lösung erschwert.
Individuelle Modifikation der UI	
Kann auf beliebiger Hardware ausgeführt werden	

Tabelle 2.5: Stärken und Schwächen der Home Assistant Plattform

Die Konfiguration von Automationen, Funktionen und Integrationen über YAML-Dateien kann einerseits als Stärke gewertet werden, da die Syntax flexibel einsetzbar ist, und andererseits als Schwäche, da die Konfigurationen schnell unübersichtlich werden und anfangs schwer zu verstehen sein kann. Der Aufwand steigt je nach Anforderungen der zu implementierenden Funktion und deren Abhängigkeiten zu Geräten und gegebenenfalls zu anderen Prozessen und Automationen.

2.4.4 Optionen der Regel- und Automatisierungserstellung

Home Assistant bietet zum Umsetzen von Regeln und Automationen²⁴ eine Option an. Dabei werden die Informationen über YAML-Dateien dargestellt. Hierbei gilt ein klares Konstrukt, welches eingehaltenen werden muss, um lauffähige Regeln und Automationen ausführen zu können. Für den Nutzer wird lediglich die Option geboten, die Automationen über eine Oberfläche zu erstellen. In dieser Form gibt ein Template die einzustellenden Möglichkeiten vor. Hierbei kann der Name und der Modus der Automation festgelegt werden. Anschließend wird ein Auslöser, beispielsweise ein Gerät oder eine Zeitangabe, definiert und entsprechend eine Bedingung zur Ausführung der Automation festgelegt. Nach den Schritten wird die eigentliche Aktion hinzugefügt, die ausgeführt werden soll, wenn die Bedingung erfüllt ist. Die generierte Automation wird als YAML-Datei gespeichert und kann ebenso über einen Editor in Rohformat editiert werden. Eine Automation über eine YAML-Datei in der Home Assistant Software gibt ein Konstrukt vor, welches mit den oben genannten Eigenschaften gefüllt werden muss. Die dafür vorgesehene Schablone sieht wie folgt aus:

```
1   automation:
2     - alias: "<NAME>"
3       trigger:
4         <TRIGGER>
5       condition:
6         - <TRIGGER_CONDITION> OR
7           <TRIGGER_CONDITION2> ...
8       action:
9         - <ACTION>
```

Code-Beispiel 2.3: Konstrukt zur Regeldefinition über Home Assistant

²⁴Erstellen von Regeln und Automationen über Home Assistant. <https://www.home-assistant.io/docs/automation/> Abgerufen am 28.05.2022

2.5 openHAB

Neben der erläuterten Home Assistant Plattform zählt openHAB zu den bekanntesten Open-Source-Systemen im Bereich des SH. Der open Home Automation Bus (openHAB) ist eine Plattform, bei der es sich um eine Softwarelösung handelt, die auf Basis der Programmiersprache Java aufgebaut ist. Die Software steht unter der Eclipse Public License und zählt zu der Rubrik der Open-Source-Software. Durch die Verwendung von Java ist die Anwendung betriebssystemunabhängig und kann auf beliebigen Betriebssystemen laufen. Ähnlich zu der vorgestellten Home Assistant Software bietet openHAB ebenso Benutzeroberflächen, sog. User Interfaces, die durch den Webbrower, Android- und iOS-Geräte unterstützt werden.

In Kombination mit Java wird bei openHAB das Open Service Gateway initiative (OSGi)-Framework für die Modularität der Software verwendet. Mit Apache Karaf wird ein Container bereitgestellt, der mit Eclipse Equinox als OSGi Laufzeitumgebung agiert. Als HTTP-Server ist Jetty in Gebrauch. Die einzelnen Frameworks werden nicht im Detail erläutert, lediglich die für das Verständnis des Konzeptes notwendigen.

Mit openHAB wird eine hochmodulare Software zur Verfügung gestellt, die durch sogenannte *Add-ons* erweitert werden kann. Durch diese wird der Plattform eine breite Palette an Funktionen geboten. Physische Geräte können in großer Anzahl mit der Plattform interagieren und verknüpft werden.²⁵

Historie

Das Smart Home Projekt begann im November 2009 und wurde von Kai Kreuzer entwickelt²⁶. Im Februar 2010 wurde das Projekt unter dem Name openHAB bekannt. Nach zweieinhalbjähriger Entwicklung, im Dezember 2012, gab Herr Kreutzer ein Statement zu der Verwendung von OSGi ab, in der er seine Entscheidung, das Framework einzusetzen, mit Begeisterung als richtig hervorgehoben hat.

„Looking back at this evolution of the project, I am perfectly sure that if I had designed openHAB as a normal Java application instead of an OSGi application, it would not have prospered as it did. It is really the choice of the software architecture that made it happen - and as a nice side effect, the community is not a pure user community as it is the case for many other Open Source projects, but it is full of engaged people who actively contribute to the project.[KREUZER 2012]“

Mit der stetig an Funktionen wachsenden Anwendung wuchs auch das Interesse der Community, an diesem Projekt mitzuwirken. 2014 betitelte Kreutzer das Jahr als Jahr des Smart Home, da die Beteiligung, das Interesse und die Entwicklung ausgesprochen zunahm und die Community immer größer wurde [KREUZER 2014]. Seither gab es unter anderem weitere architektonische Veränderungen und Verbesserungen des Systems sowie Ergänzungen um zusätzliche Protokolle zur

²⁵Einleitung zu openHAB. <https://www.openhab.org/docs/> Abgerufen am 25.04.2022

²⁶Chronologie und Blogbeiträge von Kai Kreutzer. <http://kaikreuzer.blogspot.com> Abgerufen am 28.04.2022.

Kommunikation und Anbindung verschiedenster Geräte. Im Laufe des Ausbaus wurde eine mobile Anwendung entwickelt, mit der die Remotesteuerung implementiert wurde. Die Software wird bis heute regelmäßig gepatched und von der Community unterstützt.

2.5.1 Konzept und Architektur

Die Steuerungsplattform openHAB bietet vergleichbar zu Home Assistant die Möglichkeit der multifunktionalen Verknüpfung von Geräten und Protokollen. An dieser Stelle werden ebenso mehrere Konzepte verwendet, die die Vereinheitlichung der Plattform verstärkt. Die Konzepte der openHAB Software sind in drei größere Rubriken aufgeteilt, die sich wie folgt zusammensetzen:

Die erste Rubrik sind die Dinge (Things), sogenannte Entitäten, die als physische Komponente zu einem System hinzugefügt werden können. Diese können soweit vom Hersteller ermöglicht mehrere Funktionen erfüllen, so kann zum Beispiel ein Sensor Bewegung und Temperatur erfassen. Hierbei ist zu berücksichtigen, dass die Dinge nicht immer Geräte sein müssen, sondern auch andere Informationsquellen oder Webdienste sein können. Aus Sicht des Benutzers sind sie für den Einrichtungs- und Konfigurationsprozess relevant, für den Betrieb jedoch potentiell zu vernachlässigen. Dinge können Konfigurationseigenschaften haben, die optional sein können. Solche Eigenschaften können grundlegende Informationen wie eine IP-Adresse, ein Zugriffstoken für einen Webdienst oder eine gerätespezifische Konfiguration sein, die veränderbar sind [PRANZ und SCHILLER 2018]. Mit dem Konzept der Dinge gehen zwei Unterkategorien einher:

- Kanäle (Channels): Jedes Gerät, bzw. Ding stellt Kanäle bereit, mit denen die jeweiligen Funktionen abgebildet werden. Mit der Anbindung des physischen Gerätes an die Plattform ist der Kanal eine konkrete Funktion dieser Entität. Beispielsweise kann eine Glühbirne einen Kanal für die Farbtemperatur und einen für den Farbwert besitzen. Diese stellen beide die Funktionalität der Glühbirne für das System bereit. Grundlegend sind Kanäle mit Elementen verknüpft, mit denen die virtuelle und physische Ebene verbunden wird. Sobald eine Verbindung hergestellt wird, kann ein Gerät über die Plattform angesteuert werden, indem ein Ereignis an die Entität kommuniziert wird. Wird z.B. über die Plattform eine LED-Leuchte eingeschaltet, so wird über das entsprechende Event die Lampe eingeschaltet. Einerseits ist die Verknüpfung zu einem Kanal die Vorbedingung zur vorgesehenen Kommunikation zwischen Gerät und Plattform, andererseits werden Ereignisse für Objekte nur dann gesendet, wenn diese mit mindestens einem Kanal des Dinges verknüpft sind [PRANZ und SCHILLER 2018].
- Brücken (Bridges): Die Brücke ist eine besondere Art eines Gerätes. Diese müssen dem System hinzugefügt werden, damit der Zugriff auf andere Dinge ermöglicht wird, bzw. erhalten bleibt. Ein IP-Gateway für Hausautomationssysteme, welches nicht IP-basiert funktioniert, ist ein typisches Beispiel für so eine Brücke.

An zweiter Stelle stehen die Artikel (Items). Diese Elemente stellen Funktionen dar, die von der Anwendung selbst verwendet werden. Darunter zählen hauptsächlich die Automatisierungslogik oder auch die Benutzeroberflächen. Durch Ereignisse werden Elemente verwendet, die einen veränderbaren Zustand besitzen. Elemente können in Gruppen zusammengefasst werden. Ein Gruppenelement kann

auch Mitglied einer weiteren Gruppe sein. Diese zyklischen Mitgliedschaften sind nicht verboten, werden jedoch von den Entwicklern der openHAB als nicht praktikabel dargestellt, da ansonsten die Konstellation zu komplex wird. Deshalb raten Experten von der Erzeugung zyklischer Verbindungen ab. Über Benutzeroberflächen können einzelne Gruppenelemente als alleinstehender Eintrag angezeigt werden und eine Navigation zu den jeweiligen Mitgliedern bereitstellen.

Die dritte übergreifende Rubrik ist die der Bindungen und Links (Bindings and Links). Diese können als Softwareadapter betrachtet werden, die die Dinge für Hausautomationssysteme zur Verfügung stellen [PRANZ und SCHILLER 2018]. Aufgabe der Komponente ist die Verknüpfung von Elementen mit physischen Geräten. Um dies zu ermöglichen, werden die spezifischen Kommunikationsanforderungen eines Gerätes abstrahiert. Dadurch ist eine abstrakte Behandlung der Komponenten durch das System möglich. Links stellen das Bindeglied zwischen Dingen und Gegenständen dar. Es ist die Verknüpfung von genau einem Element und einem Kanal. Elemente und Kanäle haben keine eins zu eins Beziehung zueinander. Eine Verknüpfung mit mehreren Komponenten ist auch hier möglich.

Neben den drei übergreifenden Konzepten gibt es weitere Konzepte²⁷. Diese werden im Folgenden aufgelistet:

- Seitenverzeichnis (Sitemaps): Dies beinhaltet die Konfiguration eines Seitenverzeichnisses, mit dem Schalter, Regler, Texte und vieles mehr hinzugefügt werden kann. Diese Ansichten repräsentieren den Status eines Elements. Darüber hinaus ist durch das Seitenverzeichnis die Steuerung und Überprüfung der Elemente möglich.
- Regeln (Rules): Eine Regel ist die Verknüpfung eines Ereignisses und einer Aktion. Somit wird nach einem bestimmten Ereignis eine Aktion ausgeführt. Wird z.B. ein Fenster geöffnet, soll die Heizung bis zu dessen Schließung abgeschaltet werden.
- Persistenz (Persistence): Mit der Persistenzschicht kann festgelegt werden, welche historischen Zustandsereignisse und Informationen gespeichert werden sollen. Dadurch kann nach Systemausfall der Status des Elementes wiederhergestellt werden.
- Modelle (Models): Modelle stellen baumbasierte Gruppierungen von Elementen dar. In der Baumstruktur können Elemente Orte, Punkte und Ausrüstungen sein. Beinhaltet z.B. ein Wohnzimmer als Standort zwei intelligente Geräte, Heizung und LED-Leuchte, so kann dies als Modell definiert werden. Die Heizung als Ausstattung kann wiederum weiter Informationen abbilden, bspw. die *Ist- und Soll-Temperatur*.
- Seiten (Pages): Alle Seiten und Seitenverzeichnisse kontrollieren die Zustände von Elementen und greifen auf diese zu. Mit openHAB 3 wurden weitere Möglichkeiten hinzugefügt, um benutzerdefinierte UIs, Diagramme, Grundrisse oder Layouts zu definieren.
- Skripte (Scripts): Skripte sind ähnlich zu den Regeln. Diese spezifizieren eine Aktion und kein Ereignis. Die Skripte können manuell, aber auch über andere Skripte und Regeln ausgeführt werden.

²⁷ Historie und Konzepte von openHAB. <https://www.openhab.org/docs/concepts/> Besucht am 28.04.2022

Die anfängliche Architektur der openHAB Anwendung teilt sich in drei Sektionen auf: Die Kernkomponenten (blau), das drunter liegende OSGi-Framework (grün) und die Erweiterungen (Add-Ons) (gelb).



Abbildung 2.10: Architektur der openHAB Plattform [KREUZER 2013]

Mit der wesentlichen Aktualisierung von openHAB zur architektonisch aktuellen Version openHAB 2.0 gab es auch einige Änderungen an dem grundlegenden Aufbau des Systems. Es basiert hauptsächlich auf dem Eclipse SmartHome Framework und nutzt Apache Karaf zusammen mit Eclipse Equinox, um eine OSGi Laufzeitumgebung zu erstellen. Jetty wird als HTTP Server verwendet.

*Eclipse SmartHome*²⁸ ist ein IoT Projekt, welches sich aus der ersten Version der openHAB Plattform entwickelte. Diese wurde von der Eclipse Foundation übernommen und weiterentwickelt. Es stellt das Kern-Framework für Smart Home Systeme dar und ist essentieller Bestandteil der openHAB Plattform. Das Framework ist mittlerweile archiviert und erhält keine weiteren Aktualisierungen.

*Apache Karaf*²⁹ ist eine OSGi basierte Laufzeitumgebung, über die verschiedene Komponenten mittels Container bereitgestellt werden können. Dieses Framework zählt zu denen der Apache

²⁸ Eclipse SmartHome. <https://projects.eclipse.org/projects/iot.smarthome> Abgerufen am 29.04.2022

²⁹ Apache Karaf. <https://karaf.apache.org/> Abgerufen am 29.04.2022

Software Foundation und unterliegt der Apache V2 Lizenz.

*Eclipse Equinox*³⁰ ist eine modulare Java basierte Laufzeitumgebung und eine Implementierung des OSGi Frameworks. Der Begriff der *Bundles* ist das Kernstück der OSGi-Spezifikation. Diese werden verwendet, um die Modularität für Java zu erfassen. Ein Bundle ist eine Standard-Java-JAR-Datei, dessen Manifest zusätzliches Markup enthält, welches das Bundle identifiziert und seine Abhängigkeiten angibt. Als solches ist jedes Bundle vollständig selbstbeschreibend [PRANZ und SCHILLER 2018].

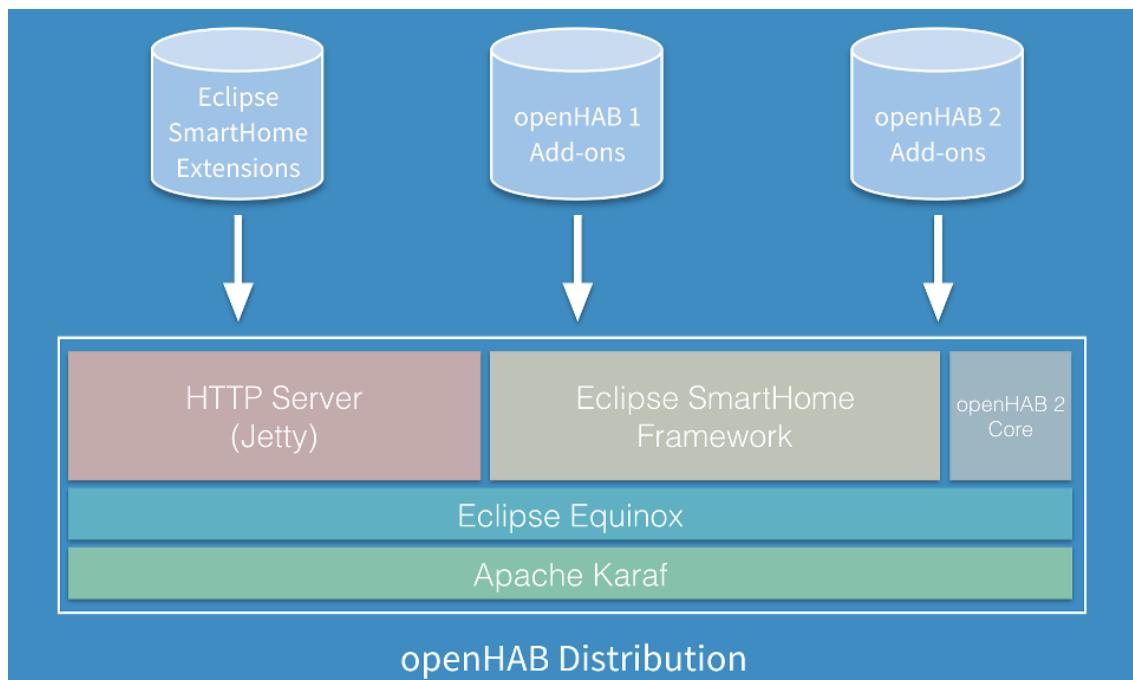


Abbildung 2.11: Architektur der openHAB 2.0 Plattform [KREUZER 2016]

Die Plattform befindet sich aktuell in der Version 3. Grundlegende Änderungen der Architektur gab es jedoch bei der Aktualisierung der Systemversion von zwei auf drei nicht, weshalb obiges Architekturnschaubild (2.11) als aktuelles gilt. Die Änderungen beinhalteten Erweiterungen des Systems, die für den Nutzer sichtbar waren, und zusätzlich Maßnahmen, um veraltete Funktionen und Konzepte abzuschalten sowie Refactorings, um die grundlegende Architektur zu vereinfachen, jedoch nicht zu ändern. Die Dokumentation³¹ zu Änderungen und Versionsupdates ist der Organisation auf GitHub zu entnehmen.

Der Open-Source-Standard *OSGi* beschreibt einen Ansatz zur Modularisierung von Software. Die Verwendung des Frameworks setzt ebenso eine Java Virtual Machine (JVM) voraus. Die Java Virtual

³⁰Eclipse Equinox. <https://www.eclipse.org/equinox/> Abgerufen am 29.04.2022

³¹Aktualisierung von openHAB 2 zu 3. <https://github.com/openhab/openhab-distro/wiki/Breaking-Changes-in-openHAB-3/139901f0ef7ee8b5b7dd480204ddf5069f030f50> Abgerufen am 03.07.2022

Machine³² ist der Teil der Java-Laufzeitumgebung, die für die Ausführung des Java-Bytecodes verantwortlich ist.

Die im Kontext des OSGi betitelten *Bundles* sind einzelne Module und bilden die kleinste Einheit der Modularisierung. Aus technischer Sicht ist das Konstrukt eine Java Archive (JAR)-Datei mit Metainformationen. Diese sind in einer Manifestdatei gespeichert. Durch die eindeutige Identifizierung von *Bundles* ist es in OSGi möglich, sie mit demselben Namen, jedoch unterschiedlicher Versionen gleichzeitig zu nutzen und auszuführen [PRANZ und SCHILLER 2018].

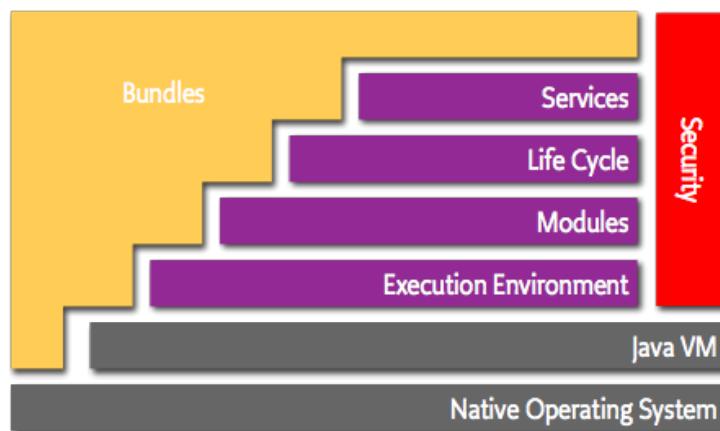


Abbildung 2.12: OSGi Schichtenarchitektur [KREUZER 2022]

Weitere Details der OSGi-Architektur³³ sind der openHAB Dokumentation zu entnehmen.

2.5.2 Ziele und Schwerpunkte

„openHAB empowering the smart home³⁴“

Das Ziel, das mit der openHAB Anwendung verfolgt wird, ist allgemein die zentrale Steuerung von Geräten innerhalb eines Wohnraumes.

„.... is an open source, technology agnostic home automation platform which runs as the center of your smart home!³⁵“

2.5.3 Stärken und Schwächen

Die Stärken, die openHAB mit sich bringt, werden schon in den Dokumentationen der Plattform aufgegriffen und bei der Nutzung der Anwendung schnell deutlich. Mit der Plattform können eine Vielzahl von Geräten und Systemen integriert werden, die miteinander kommunizieren. In der

³² Beschreibung der JVM. <https://www.javatpoint.com/jvm-java-virtual-machine> Besucht am 01.05.2022

³³ Erläuterung von OSGi. <https://www.openhab.org/docs/developer/osgi/osgi.html> Besucht am 01.05.2022

³⁴ Log und Label von openHAB <https://www.openhab.org/> Besucht am 01.05.2022

³⁵ Übersicht der openHAB Dokumentation. <https://www.openhab.org/docs/> Besucht am 01.05.2022

einzigsten von openHAB gegebenen Lösung können Heimautomatisierungssysteme, intelligente Geräte und andere Technologien, darunter beispielsweise Kommunikationsprotokolle, integriert werden [KREUZER 2020]. Durch die in dem letzten großen Update der Plattform hinzukommende einheitliche Benutzeroberfläche, die die Bedienung von Geräten und Systemen vereinfacht, ist die Integration von neuen Geräten einfacher. Mit dem Ansatz von Automatisierungsregeln können alle Geräte bei Bedarf miteinander kommunizieren. Das Aufstellen von solchen Regeln ist herstellerunabhängig und macht die Automatisierung flexibler. Der dritte und letzte Punkt der Stärken ist die allgemeine Vielfältigkeit und Flexibilität der Plattform. Hiermit ist im Allgemeinen die Umsetzung von Use Cases und Automatisierungen gemeint.

Zu den Schwächen zählt hauptsächlich die vielschichtige und weitgestrickte Architektur. Das System an sich und der Umgang damit stellt sich als komplex dar. Das Aufsetzen der Plattform und das Verstehen der Konzepte und Möglichkeiten sind sehr zeitintensiv. Dies spiegelt die Dokumentation der Software wieder [KREUZER 2020].

Eine zusätzliche Schwäche ist die Weiterentwicklung des Systems. Diese findet langsam und nicht konstant statt. Auslöser dafür ist die Architektur und deren Abhängigkeiten zu weiteren Frameworks und Bibliotheken. Auch nimmt der Prozess bis zur Genehmigung von Weiterentwicklungen viel Zeit in Anspruch. Dies ist möglicherweise mit ein Grund, weshalb neue Versionen und Updates nur unregelmäßig zu beobachten sind.

2.5.4 Optionen der Regel- und Automatisierungserstellung

Im Vergleich zu Home Assistant bietet openHAB zur Definition und Erstellung von Regeln mehrere Möglichkeiten an, die über deren Oberfläche definiert werden können. Die Konfiguration solcher Regeln kann jedoch mit unterschiedlichen Methoden umgesetzt werden. Zum einen mit einem visualisierten Baukasten Werkzeug Namens *Blockly*³⁶ und zum anderen über eine Domain Specific Language (DSL), im Deutschen eine domänenspezifische Sprache³⁷. openHAB verwendet für die Syntax der Regeldefinition Xbase³⁸ und darauf aufbauend Xtend³⁹. Diese DSL gibt ein Konstrukt vor, mit dem Regeln definiert werden können. Dies sieht wie folgt aus:

```

1      rule "<RULE_NAME>"  

2      when  

3          <TRIGGER_CONDITION> [or <TRIGGER_CONDITION2> [or ...]]  

4      then  

5          <SCRIPT_BLOCK>  

6      end

```

Code-Beispiel 2.4: Konstrukt zur Regeldefinition über Xbase

³⁶Ein visueller Code Editor von Google. <https://developers.google.com/blockly> Besucht am 29.05.2022

³⁷Erklärung der domänenspezifischen Sprache. <https://martinfowler.com/dsl.html> Besucht am 29.05.2022

³⁸Erläuterung des Xbase Frameworks. <https://www.eclipse.org/Xtext/index.html> Besucht am 29.05.2022

³⁹Spezieller Javadialekt. <https://www.eclipse.org/xtend/index.html> Besucht am 29.05.2022

Jede Regel bekommt einen eindeutigen Namen. Danach muss eine Bedingung implementiert werden, bei deren Eintreffen eine bestimmte, beliebige Aktion ausgelöst wird. Die Logik der Regel wird über den Skript-Block eingesetzt. Für die Syntax der Regellogik wird *Xtend* verwendet.

2.6 Vergleich von Home Assistant und openHAB

Nach der Erläuterung der beiden Softwarelösungen im Bereich der Smart Home Plattformen werden diese abschließend gegenübergestellt, um weitere Aspekte miteinander zu vergleichen. Grundlage dafür sind persönliche Erfahrungen, als auch Eindrücke von Nutzern und Experten. Die Tabellen (B.1 & B.2) zum Vergleich der Plattformen ist dem Anhang (B) zu entnehmen.

Zusammenfassend zu dieser tabellarischen Gegenüberstellung ist zu sagen, dass beide Plattformen Vorzüge aufweisen. Beide Ansätze bieten eine gute Grundlage, die Automatisierungen in dem eigenen Gebäude voranzutreiben. Die Entscheidung, welches Tool verwendet werden soll, liegt in der Verantwortung des Nutzers. Übergreifend sind die beiden Open-Source-Lösungen eine gute Alternative zu kommerziellen Produkten und Lösungen, sofern der Nutzer herstellerunabhängig interagieren möchte. Dennoch ist es für Nutzer und Entwickler mit einem hohen Aufwand verbunden, Automationen und Regeln zu definieren, da Werkzeuge eingesetzt werden, die nicht dem Standard der verwendeten Programmiersprache, beispielsweise von Java, entsprechen. Dadurch muss zuerst die Funktionsweise des Frameworks verstanden werden, um anschließend Automationen und Regeln definieren und anwenden zu können. Beide Lösungen versuchen diese Herausforderung anzugehen, indem über eine Weboberfläche die Regeln konfiguriert und erstellt werden können. Home Assistant nutzt hierfür, wie bereits erläutert, das YAML-Dateiformat. Im Gegenzug dazu verwendet openHAB ein Framework, das das Konstrukt der Regeldefinition starr vorgibt. Alternativ können Regeln auch mittels *Blockly* realisiert werden.

Beide Softwarelösungen bieten einen unterschiedlichen Ansatz zur Regeldefinition, jedoch ist die Flexibilität auf integrierte Komponenten beschränkt. Das bedeutet, dass nur vom System abbildbare Komponenten, Integrationen und Verknüpfungen verwendet werden können. Gibt es beispielsweise für eine Komponente oder ein Gerät keine Integration oder kein Add-on, so muss dieses erst im Quellcode als Packet implementiert und angeboten werden, bevor der Anwender dieses benutzen kann.

Die soeben erläuterten Grundlagen geben einen Einblick in eine Domäne des IoT. Mit den Kommunikationsprotokollen werden Standards und derzeit oft verwendete Technologien beschrieben. Das MQTT-Protokoll wird im Rahmen dieser Arbeit von Anfang an verwendet. Die Erläuterung von zwei Softwareplattformen gibt einen Überblick darüber, welche Lösungen bereits vorhanden sind und wie sich deren Aufbau gestaltet. Diese Information ist notwendig, um im weiteren Verlauf die Konzeption als auch die anschließende Analyse nachvollziehen zu können.

Kapitel 3

Stand der Technik

Zur Analyse des aktuellen Standes der Technik und Forschung in Bezug auf die Konzeption von Software-Lösungen, mit denen die formalisierten Interaktionen der Softwareentwickler vereinfacht werden können, erfolgt in diesem Kapitel ein systematisches Literaturreview. Die Literaturprüfung wird gemäß den Richtlinien, die in der Publikation von [KITCHENHAM und CHARTERS 2007] vorgeschlagen werden, durchgeführt.

3.1 Systematisches Literaturreview

Die Thematik des systematischen Literaturreviews wurde bereits in den einleitenden Kapiteln erwähnt. Dieser Abschnitt widmet sich ausschließlich der Anwendung der Richtlinien und dem daraus abgeleiteten Stand der Technik abhängig zu dem in der Arbeit behandelten Thema.

3.1.1 Ziele des Systematischen Literaturreviews

Das Ziel dieses systematischen Literaturreviews ist es, die aktuellen Fortschritte von Smart Home Plattformen und Gateways in Richtung der entwicklerseitigen Benutzerfreundlichkeit zu recherchieren. Dabei liegt der Schwerpunkt auf der Usability und der einfachen Handhabung der formalisierten Interaktionen der Softwareentwickler. Es gilt zu analysieren, ob es in diesem Themenbereich bereits Publikationen und Forschungen gibt und welche Entscheidungen getroffen werden müssen, um die Weiterentwicklung eines Systems je nach hinzukommender Funktionalität oder auch Bedingung übersichtlich zu halten. Die Ergebnisse dieses systematischen Literaturreviews sollen als Grundlage der Konzeption einer solchen Plattform dienen und mit einfließen.

3.1.2 Suchstrategie- und anfragen

Dieser Abschnitt beschreibt die Suchstrategie und die Anfragen zu dem systematischen Literaturreview. Hierbei wird erläutert, anhand welcher Kriterien die Literatur ausgewählt wird.

In den ersten Schritten werden anhand der Forschungsfragen in Kapitel (1.2) die Stichpunkte aufgegriffen und als Suchterm formuliert. Die daraus resultierenden Suchterme, die der tabellarischen Darstellung (3.1) zu entnehmen sind, werden in diversen wissenschaftlichen Fachdatenbanken

recherchiert und analysiert. Die Ergebnisse werden nach ihrem Titel und der Zusammenfassung sortiert. Gibt es Publikationen mit vergleichbaren Inhalten, so sind diese in weiteren Schritten näher zu betrachten. Damit die Literaturrecherche weitere Ergebnisse erzielt, wird beim studieren der Publikationen die Schneeballsuche angewendet. Dabei wird in den Quellen der jeweiligen Literatur auf weitere Verweise geschaut, die ebenso potentielle Inhalte bearbeiten. Die Kombination beider Literaturergebnisse bilden die Grundlage der zu analysierenden Quellen und geben so den Stand der Technik wieder.

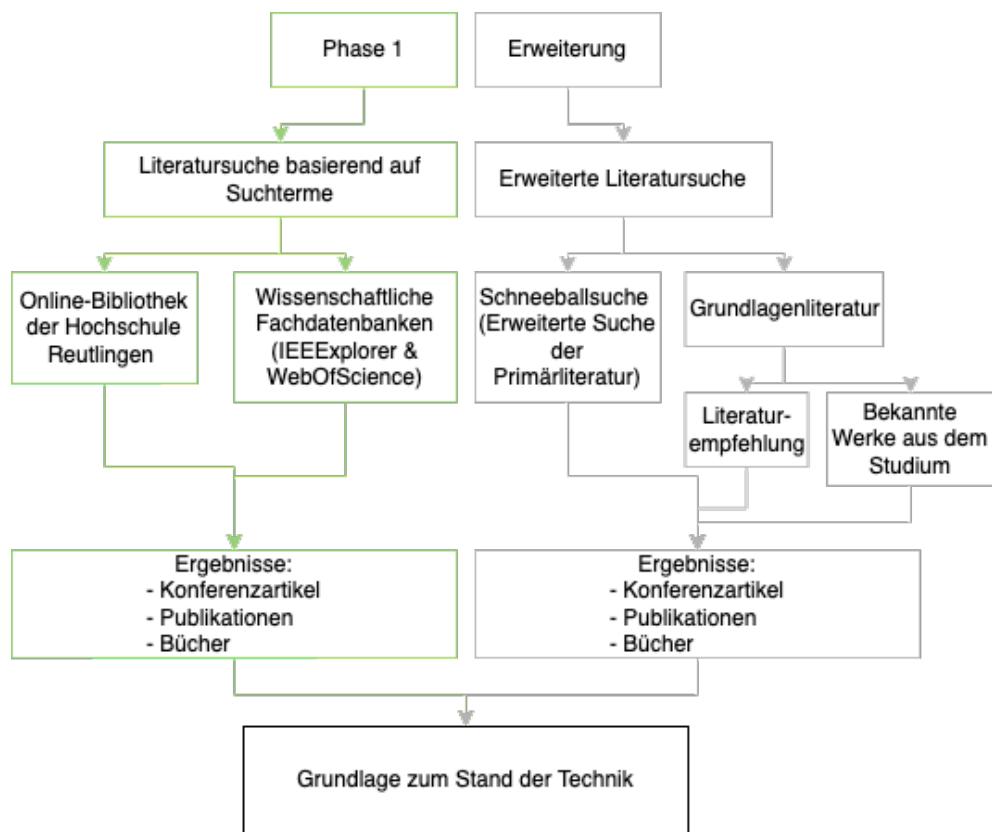


Abbildung 3.1: Strategie der Literatursuche

Die generierten Suchterme, die daraus resultierenden Literaturergebnisse und der damit einhergehende Suchverlauf in Auszügen ist zur Nachvollziehbarkeit tabellarisch aufgeführt:

Suchanfrage	Datum	Filter	Plattform	Ergebnisse	Gesehene	Relevant
formalized interactions software development architecture	11.04.2022, 03.05.2022	Nein	IEEEExplorer, WebOfS-science	13, 26	6, 4	0, 0
usability AND formalized interactions AND architecture	11.04.2022, 03.05.2022	Nein	IEEEExplorer, WebOfS-science	3, 7	1, 3	0, 1
usability AND formalized interactions AND architecture AND smart home	01.05.2022	Nein	IEEEExplorer, WebOfS-science	0, 0	0, 0	0, 0
usability AND formalized interactions AND software development AND architecture	02.05.2022	Nein	IEEEExplorer, WebOfS-science	1, 1	1, 1	0, 1
usability AND architecture AND gateway AND smart home	06.05.2022	Nein	IEEEExplorer, WebOfS-science	2, 4	1, 1	1, 1
usability AND formalized interaction AND (architecture OR smart home OR software developer OR gateway)	09.05.2022	Nein	IEEEExplorer, WebOfS-science	3, 10	1, 2	0, 0
usability AND architecture AND smart home AND (formalized interaction OR software developer OR iot)	09.05.2022	Nein	IEEEExplorer, WebOfS-science	13, 22	3, 3	0, 0

Tabelle 3.1: Suchprotokoll des Systematischen Literaturreviews

3.1.3 Datenextraktion und Synthese

Zu der zielgerichteten Datenextraktion werden in Anlehnung an die Richtlinien des systematischen Literaturreviews die folgenden Einschluss- und Ausschlusskriterien definiert, die dabei helfen, die relevanten Publikationen zu finden:

Einschluss-Kriterien

-
- 1 Die Literatur ist in den wissenschaftlichen Fachdatenbanken veröffentlicht, darunter: WebOfScience, IEEEExplorer, SpringerLink, Elsevier, Addison-Wesley, dpunkt-Verlag, ACM und Google Scholar
 - 2 Der Beitrag wurde nach 2010 veröffentlicht
 - 3 Die Veröffentlichung ist in deutscher oder englischer Sprache
-

Ausschluss-Kriterien

-
- 1 Die Veröffentlichung beinhaltet, bzw. behandelt nicht die Schlagworte usability, architecture, formalized interaction oder iot
 - 2 Die Publikation gehört zu der Literatur der Grauzone
 - 3 Die Veröffentlichung hat weniger als 5 Zitationen
 - 4 Die Literatur hat weniger als 5 Seiten Inhalt
-

Tabelle 3.2: Einschluss- und Ausschlusskriterien des systematischen Literaturreviews

Anhand der Forschungsfrage, den Suchterminen und den Einschluss- und Ausschlusskriterien zu der Quellenrecherche werden während der Anwendung des systematischen Literaturreview-Templates und deren Richtlinien die erzielten Ergebnisse synthetisiert und zusammengefasst. Im Rahmen dieser Forschungsfrage und der Auslegung der Suchterme, die der Tabelle (3.1) zu entnehmen sind, ergab sich nicht die Menge an Literatur, die notwendig ist, um ein umfangreiches Literaturreview im Stil des Templates durchzuführen. Demnach ist kein vollständiges systematisches Literaturreview möglich. Die wenigen relevanten Ergebnisse des Literaturreviews geben einen Einblick in Bereiche, die als Gedankenanstöß und zum Transferieren von Wissen, Gedanken und Ideen geeignet sind.

Das Resultat des durchgeführten Literaturreviews beweist, dass es in diesem Bereich bisher wenige Publikationen gibt und sich dadurch eine neue Nische bilden könnte. Um dennoch eine fundierte wissenschaftliche Grundlage zu repräsentieren, wird in folgendem Abschnitt auf Referenzen und Literatur eingegangen, die Teile der Forschungsfrage abdecken und mehr als Gedankenanstöß anzusehen sind, beziehungsweise auch einen partiellen Einblick in den Stand der Technik bietet.

3.2 Zusammenfassung

Speziell auf die Forschungsfrage in Kapitel (1.2) gibt es nach dem heutigen Stand der Technik dazu keine entsprechende Fachliteratur, welche den Gedanken aufgreift und behandelt. Bei der systematischen Literaturrecherche tauchen als Ergebnis interessante Anhaltspunkte auf, die übertragen in den Kontext dieser Arbeit hilfreiche Erkenntnisse beitragen und Impulse für die Konzeption geben konnten.

3.2.1 Publikationen

Im Folgenden werden diese aufgegriffen und zusammengefasst, sodass ein Einblick in den Prozess der Informationserhebung, der Sammlung von Erfahrungswerten und Ideen gewährleistet wird.

Design and Realization of a Framework for Human-System Interaction in Smart Homes

In dem Artikel von [WU und FU 2012] wird zu Anfang die Beziehung zwischen Benutzern, Räumlichkeiten und Diensten analysiert. Basierend auf den daraus gewonnenen Erkenntnissen wird ein Framework und ein entsprechender Algorithmus vorgestellt, welcher die Interaktionsbeziehungen modelliert. Aufgrund dieser Ergebnisse wird ein Framework entwickelt, welches die Interaktionsanforderungen abdeckt. Hauptmerkmale waren dabei Komfort, Bequemlichkeit und Sicherheit. Zur abschließenden Überprüfung des Designkonzeptes und der Implementierung wurden Probanden zum Testen der Anwendung ausgewählt und anschließend ein Interview durchgeführt. Das Evaluierungsergebnis zeigt, dass das Framework eine gute Einführung in die Verbesserung der Mensch-System-Interaktionen darstellt.

Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes

Der Artikel von [KIM u. a. 2012] erarbeitet einen Vorschlag einer ganzheitlichen und erweiterbaren Softwarearchitektur, welche Dienste und heterogene protokoll- und herstellerspezifische Geräte nahtlos integrieren lässt und Sicherheit über das Internet gewährleistet. Grundlegend wird hierbei auf das OSGi Framework gesetzt, wodurch die semantische Interoperabilität hervorgehoben wird. Dies bezeichnet die Fähigkeit, neue Anwendungen und Treiber zur Laufzeit in das bereitgestellte System zu integrieren [KIM u. a. 2012]. Zusätzlich zu dem System wird im Rahmen dieser Publikation ein Zugangskontrollmodell für spezielle Smart Home Szenarien integriert. Zur Beweisführung wird das Konzept anhand von semantischen Erkennungen von Heimgeräten zur Laufzeit demonstriert. Dafür werden mehrere Protokolle, darunter X10, ZigBee und Insteon, in einen realen Test integriert. Die Arbeit behandelt die folgenden Schwerpunkte [KIM u. a. 2012]:

- Analyse einer Reihe von Heimnetzprotokollen hinsichtlich ihrer Erkennungs- und Integrationsanforderungen.
- Eine erweiterbare Home-Gateway-Architektur, die es ermöglicht, heterogene Geräte während der Laufzeit flexibel zu installieren, zu verwalten und darauf zuzugreifen.
- Ein neuartiger Zugangskontrollmechanismus speziell für Smart-Home-Systeme.
- Die Umsetzung des vorgeschlagenen Konzeptes, indem gezeigt wird, wie verschiedene Geräte integriert und von Endbenutzern aufgerufen werden können.

Laut den Ergebnissen unterstützt die Architektur und die damit eingesetzte semantische Abstraktionsschicht die Anwendungsentwicklung erheblich.

Mit der Zugriffskontrollrichtlinie wird den Hausbesitzern eine stabile Kontrolle darüber gegeben, mit der die Benutzer auf die smarten Geräte zugreifen können. Das Resultat dieser Arbeit scheint zu zeigen, dass damit die Barriere für Smart Home Systeme gesenkt wird.

Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation

Dieser Beitrag von [VIANI u. a. 2013] diskutiert die aktuellen Trends von drahtlosen Architekturen für Anwendungen im Smart Home Bereich. Aus der Diskussion wurden Vorteile erarbeitet, die anschließend über die Verwendung der drahtlosen Architektur analysiert wurden. Schwerpunkt dabei lag jedoch auf der Schätzung des Benutzerverhaltens.

Allgemein behandelt der Artikel die Vorstellung einer drahtlosen Architektur für intelligentes Energiemanagement und die Überwachung in Bezug auf ältere Menschen, indem die Anwesenheit, die Bewegung und das Verhalten der Bewohner in Altersresidenzen analysiert werden [VIANI u. a. 2013].

Interessant dabei ist das entstandene Konzept der konkreten Softwarearchitektur und die Übertragbarkeit auf Anwendungsfälle im Smart Home.

My House, My Rules: A Private-by-Design Smart Home Platform

Dieses Whitepaper von [ZAVALYSHYN u. a. 2020] stellt eine *Private-by-Design-IoT-Platform* für Smart Home Umgebungen vor. Mit dem Konzept wird eine typische Architektur für bestehende IoT Plattformen als Grundlage verwendet, die über ein alternatives Design mehr Sicherheit und Kontrolle für den Hauseigentümer bietet. Genutzt wird dabei die von Intel entwickelte Software Guard Extentions (SGX)¹. Diese Erweiterung ermöglicht es, eine intuitive Sicherheitsabstraktion einzuführen, die den unbefugten Zugriff auf Daten durch nicht vertrauenswürdige IoT Cloud-Anbieter verhindert. Aus dem Konzept entstand ein Prototyp, der anschließend evaluiert wurde. Dafür wurde eine quantitative Forschung mit mehr als 40 Probanden durchgeführt, die den Prototyp verwendet und anschließend bewertet haben. Die Mehrzahl der Teilnehmer/-innen der Feldstudie hielten die Softwareplattform als benutzerfreundlich und die unterstützenden Richtlinien durch die Sicherheitsabstraktion für nützlich [ZAVALYSHYN u. a. 2020]. Mit den Richtlinien kann die Privatsphäre der Anwender in ihrem Wohnraum hinreichend geschützt werden. Der Sicherheitsmonitor der Software ermöglicht Endbenutzern eine granulare Kontrolle und Überwachung der Datenflüsse, die durch die IoT Geräte generiert werden, sowie die Verhinderung von potentiellen Datenschutzverletzungen der Hersteller durch die Verwendung einer Datenschutzrichtlinie.

Fast-prototyping Approach to Design and Validate Architectures for Smart Home

Inhalt des Artikels von [MONTANARO u. a. 2021] ist das Entwickeln eines komplexen Smart Home Systems. Hintergrund dafür ist die kontinuierliche Entwicklung und Kommerzialisierung sämtlicher IoT Geräte und die damit einhergehende Änderung oder Anpassung der Nutzeranforderungen. Dadurch benötigt die Community eine schnelle Lösung, bzw. einen Prototypen, um die Anforderungen der Nutzer erfüllen und schnell auf entstehende Bedürfnisse reagieren zu können.

Grundlage für die Entwicklung der Plattform ist eine aktuelle und solide Studie, die ebenso im

¹Eine hardwarebasierte Verschlüsselung von Speicherinhalten, die bestimmten Programmiercode und Daten im Speicher isoliert. <https://www.intel.de/content/www/de/de/architecture-and-technology/software-guard-extensions.html> Abgerufen am 15.05.2022.

Rahmen des Artikels durchgeführt wurde. Die Benutzeranforderungen wurden aus der Studie extrahiert und sind bei der Planung des Konzeptes mit eingeflossen. Bestandteil dieser Arbeit ist unter anderem die Verwendung von Node-RED² und dem MQTT Kommunikationsprotokoll.

Die in der Publikation vereinfachten formalisierten Interaktionen werden speziell dem Anwender gegenüber durch Node-RED visualisiert und somit die komplexe Logik vereinfacht dargestellt. Die Idee der Verwendung von Node-RED ist ein innovativer Gedanke der formalisierten Interaktionen und kann dem Entwickler programmatisch Schritte erleichtern. Eine gewisse Komplexität ist dabei nicht zu verhindern, da die Funktionsweise des Frameworks selbst verstanden werden muss. Eine Optimierung in weiteren Forschungsschritten ist möglich. Mithilfe des Systems können im Rahmen von praktischen Arbeiten im Forschungs- und Bildungsbereich kleinere Anwendungsfälle binnen weniger Schritte umgesetzt und getestet werden. Dadurch kann der Lerneffekt im Umgang mit Geräten im Smart Home optimiert werden.

Alle oben aufgeführten Artikel und Forschungsarbeiten sind keine explizite Referenz, bzw. stehen in keinem Zusammenhang zu der in dieser Arbeit gestellten Forschungsfrage. D.h. sie sind nicht übertragbar auf die Inhalte dieser Arbeit, sondern es konnten nur Fragmente oder Impulse daraus verwendet werden, um einen Einblick in den Stand der Forschung zu erhalten.

Mittels weiteren Experteninterviews wird deutlich, dass viele mit der Thematik des Smart Home vertraut sind, jedoch überwiegend nur als Benutzer bestehender Plattformen und Geräte gelten. Die dadurch erlangten Informationen sind nicht repräsentativ, lediglich eine gewonnene Erkenntnis im Rahmen dieser Arbeit.

Um einen Gesamtüberblick bezüglich des Technikstandes rund um Smart Home zu vermitteln, wird dieser im Folgenden aus einer etwas pragmatischeren Sichtweise erläutert.

3.2.2 Stand der Technik aus Nutzer- und Produktsicht

Der aktuelle Markt bietet bereits viele Möglichkeiten und Alternativen zur Umsetzung eines intelligenten Zuhause. Darunter fallen z.B. kommerzielle in sich geschlossene Systeme, die ausschließlich dem Nutzer als Anwendung verkauft werden oder einzelne intelligente Geräte, die nicht zwangsläufig eine zentrale Steuerungsplattform benötigen. Ebenso gibt es Plattformen, wie bspw. Home Assistant und openHAB, die eine gewisse Technikaffinität erfordern und voraussetzen, um diese nach belieben einzusetzen und konfigurieren zu können. Große Unternehmen, darunter Bosch, Telekom, Siemens, Samsung bieten Geräte und Softwarelösungen an, bei denen Komponenten des gleichen Herstellers kombinierbar sind. Unterschiedliche Geräte verschiedener Anbieter lassen sich nicht so gut verknüpfen wie bei den Open-Source-Plattformen. Sie geben keinen Einblick in ihre Software und deren Umsetzung, daher sind diese zum Vergleich nicht weiter zu berücksichtigen.

Neben den bekanntesten, frei verfügbaren Systemen openHAB und Home Assistant gibt es weitere

²Ein von IBM entwickeltes grafisches Entwicklungswerkzeug mit Baukastenprinzip. Funktionsbausteine können per Verbindungen miteinander verknüpft und als Prozess bearbeitet werden. <https://nodered.org/> Abgerufen am 15.05.2022.

Lösungen, die eine Plattform zur Verfügung stellen³:

- OpenMotics
- Jeedom
- ioBroker
- AGO Control
- Domoticz
- Homebridge.io

Die Auflistung gibt nicht alle am Markt verfügbaren Produkte wieder, lediglich eine Auswahl, der am häufigsten benutzt werden. Durch die Vielzahl der Angebote und dem wachsenden Mehrwert, den eine solche Plattform bietet, nimmt die Forschung und Entwicklung in dem Bereich des IoT zu.

Für das Abbilden und Verwalten von Prozessen und Automatisierungen gibt es sogenannte Regelwerke. Sie sind dafür entwickelt, bestimmte Regeln, Prozesse und Sachverhalte zu definieren und zu durchlaufen. Einsatz finden diese Frameworks in Business-Rule-Management-System (BRMS)en, die dazu beitragen, ein auf Geschäftsregeln basierendes Programm zu entwickeln. Ebenso werden solche Systeme bei Smart Home Plattformen eingesetzt. Durch die Definition von Regel und Automatisierungen können wiederkehrende Prozesse in einem smarten Büro oder Zuhause automatisiert werden, die dem Nutzer Aufgaben abnehmen.

Es gibt eine Vielzahl von Anbietern, die solche Systeme und Frameworks in unterschiedlicher Implementierung bereitstellen. Die im Java-Umfeld bekanntesten sind unter anderem Drools, OpenL Tablets, Easy Rules und RuleBook⁴.

Im folgenden Kapitel ist eine Marktanalyse (siehe Abschnitt 4.1) im Bereich Smart Home dargestellt, die auf die Fortschritte und Prognosen der nächsten Jahre eingeht.

³Aufstellungen von verfügbaren Open Source Smart Home Plattformen im Jahr 2020. <https://ubidots.com/blog/open-source-home-automation/> Abgerufen am 16.05.2022

⁴Regelwerke im Java-Umfeld. <https://www.baeldung.com/java-rule-engines> Besucht am 16.05.2022

Kapitel 4

Anforderungsanalyse

Dieses Kapitel befasst sich im Allgemeinen mit der Anforderungsanalyse und -erhebung. Hierbei wird eine Marktanalyse präsentiert, um das Potential rundum Smart Home aufzuzeigen und die entstehenden oder bereits bestehenden Anforderungen aufzuzeigen. Die Analyse ist mit repräsentativen Statistiken, Studien und Umfragen belegt. Hauptsächlich wird im Rahmen der Anforderungserhebung auf die Methodiken und Techniken eingegangen, die verwendet werden, um Anforderungen zu identifizieren. Diese dienen als Grundlage für die Konzeption der Steuerzentrale. Bestandteile der Anforderungserhebung sind unter anderem zentrale Prozesse des Requirements Engineering, ein *user-centered Design* (nutzerzentriertes Design), eine *Target Group Analysis* (Zielgruppenanalyse), und die Durchführung von Experteninterviews. Diese Interviews sind nicht repräsentativ und dienen lediglich der weiträumigeren Informationsgewinnung.

Vorab wird sichergestellt, dass im Rahmen des benutzerzentrierten Designs der Softwareentwickler als Nutzer und Anwender im Vordergrund steht, da dieser die Plattform betreibt, bzw. für die Erweiterung der Software als auch für die Anpassungen auf die eigenen Anwendungsfälle zuständig ist.

Damit ein Eindruck entsteht, welches Marktpotential Smart Home Anwendungen haben und welche Anforderungen somit verbunden sind, wird basierend auf bestehenden Studien, Statistiken und Umfragen eine Marktanalyse durchgeführt.

4.1 Marktanalyse

Der Marktanteil rundum Smart Home wächst weiter an, sei es in Bezug auf die Entwicklung von neuen intelligenten Geräten, die Massentauglichkeit von Geräten oder die stetig wachsende Abdeckung von Anwendungsfällen und Übernahme von Aufgaben und Prozessen. Durch die Vielzahl an Produktanbietern und diversen Kommunikationsmöglichkeiten ist es schwierig, eine Lösung für alle Alternativen und Produktausprägungen anzubieten. Hersteller versuchen mit ihrer Produktpalette ihr eigenes Ökosystem im Bereich Smart Home zu erstellen, um die Nutzer abhängig zu machen. Der repräsentativen Studie von Deloitte zufolge ist jedoch eine Insellösung bei den Nutzern in Deutschland nicht gefragt [WAGNER u. a. 2018]. Befragt wurden 2000 Konsumenten zwischen 19 und 75 Jahren. Einem geringen Anteil von 22 Prozent der Befragten ist die Erweiterbarkeit des Systems mit

Produkten anderer Hersteller weniger, bzw. nicht wichtig. Im Gegensatz dazu empfinden 43 Prozent der Befragten die Erweiterbarkeit als wichtig und 28 Prozent als sehr wichtig [WAGNER u. a. 2018]. Demnach sollten die Hersteller eine flexiblere Einsetzbarkeit gewährleisten, damit solche Systeme den Marktdurchbruch erlangen. Allerdings wird dadurch die Weiterentwicklung von Plattformen komplexer und umfangreicher, was sich wiederum als Nachteil erweist. Beispielsweise sind die am weitverbreitetsten Open-Source-Plattformen, openHAB und Home Assistant, vielschichtig und bilden ein großes System mit zunehmend integrierbaren Geräten ab, deren Funktionsumfang stetig steigt.

4.1.1 Allgemeine Marktsituation und Marktprognose

Derzeit gibt es viele Anbieter für intelligente Produkte. Diese bieten zum einen einzelne Geräte an, die in beliebige Plattformen integriert werden können und zum anderen ein eigenes Ökosystem, sofern der Anwender mehrere Produkte eines Anbieters nutzen möchte. Dennoch ist in den meisten Fällen die Konfiguration der Geräte nur auf den hauseigenen Plattformen möglich. Somit kann der Nutzer nicht alle Komponenten ausschließlich über eine Plattform konfigurieren und steuern.

Eine repräsentative Umfrage der Statista Global Consumer Survey (SGCS) mit 1384 Teilnehmern, die im April 2022 veröffentlicht wurde, zeigt, welche Anbieter in Deutschland am populärsten sind, bzw. am häufigsten von den Nutzern gekauft werden. An oberster Stelle steht Philips und Samsung mit jeweils 25 Prozent und an dritter Stelle Bosch mit 23 Prozent. Weitere Anbieter können dem Diagramm im Anhang (siehe C) entnommen werden. Es sind jedoch nicht alle Hersteller und Anbieter aufgelistet.

Der aktuelle Markt für intelligente Produkte ist in sechs primäre Segmente aufgeteilt, die jeweils andere Anwendungsfälle abdecken [LASQUETY-REYES 2021]:

- Kontrolle und Konnektivität: Gateways die alle Geräte jeglicher Segmente kontrollieren, intelligente Lautsprecher mit dem Fokus zur Kontrolle und der digitalen Unterstützung und bspw. intelligente Steckdosen
- Intelligente Geräte (Smart Appliances): Kühlschrank, Waschmaschine und Geschirrspüler, Kaffeemaschine, Mikrowelle und Staubsaugerroboter
- Sicherheit: Bewegungs-, Wasser- und Rauchmelder, Kameras und Türschlösser
- Heimunterhaltung (Home Entertainment): Fernseher, Entertainment-Systeme
- Komfort und Licht: Intelligente LEDs, Fenster- und Tür-Sensoren etc.
- Energiemanagement: Thermostate und Regler, Luftqualitätsmesser etc.

Die aufgelisteten Segmente werden von vielen Herstellern bedient. Der folgenden Abbildung sind die repräsentativen Schlüsselanbieter zu entnehmen:

Representative Smart Home key players by type and segment¹

	Control and Connectivity	Comfort and Lighting	Security	Home Entertainment	Energy Management	Smart Appliances
Dedicated segment companies	Control4 FIBARO® INSTEON® LOXONE GIRA eQ3	LEDVANCE LIFX moodnode COMFLYLIGHT	ALARM.COM CHUANGO® canary ADT Security August vivint.SmartHome™	SONOS PURE ROKU D. DEFINITIVE TECHNOLOGY	tado° ecobee climate™ nest netatmo	ECOVACS ROBOTICS iRobot® neato robotics®
Players entering the market from foreign industries	HomeKit Baidu amazon echo MI belkin NETGEAR	T... link somfy. LEEDARSON SAMSUNG SmartThings™	AT&T ASSA ABLOY SCHLAGE Gigaset	B&O BANG & OLUFSEN apple tv logitech BOSE DENON	hive Danfoss innogy BOSCH Honeywell Schneider Electric	B/S/H/ Haier LG Whirlpool CORPORATION

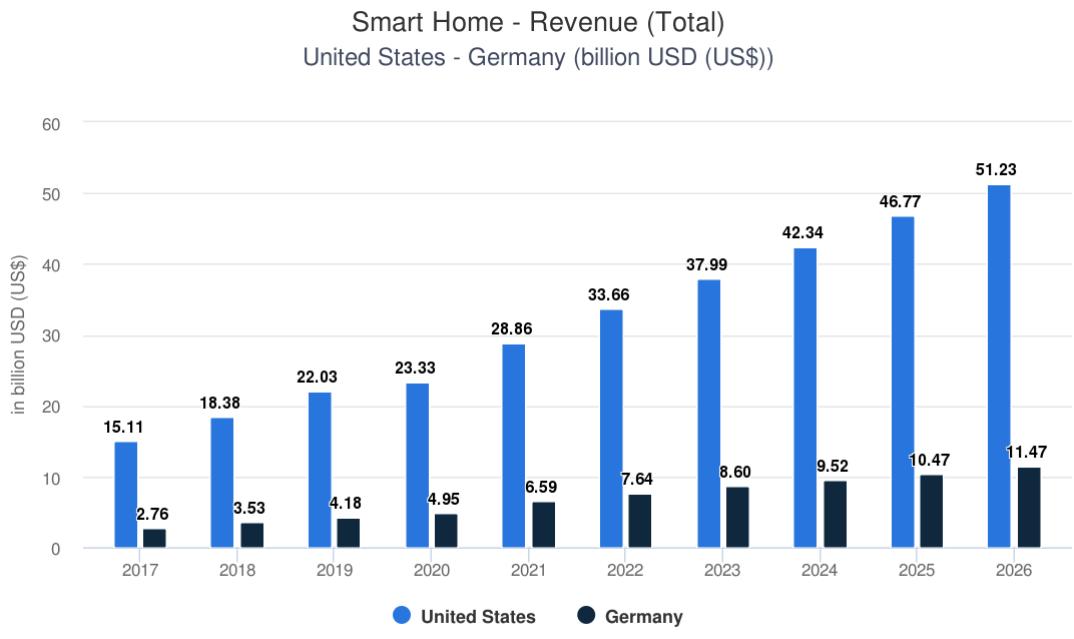
11 1: Key player overview does not represent the entire market landscape
Sources: Statista Digital Market Outlook, as of April 2021

Abbildung 4.1: Übersicht der repräsentativen Schlüsselanbieter [LASQUETY-REYES 2021]

Die Übersicht deckt jedoch nicht alle Anbieter ab und spezialisiert sich in diesem Fall auf die bekanntesten und die am Markt etablierten.

Laut den von Statista veröffentlichten Daten war die USA mit einem Umsatz von 28,86 Milliarden US-Dollar der größte Smart Home Markt im Jahr 2021, wogegen Deutschland einen Umsatz von 6,59 Milliarden US-Dollar erzielte. Zu berücksichtigen sind dabei jedoch die demographische Lage als auch die Bevölkerungsdichte. Diese Aufstellung steht in keinem direkten Vergleich und dient lediglich zur Veranschaulichung und zur Unterscheidung der Marktanteile. Deutlich wird dabei trotzdem das ähnlich prozentual ansteigende Marktwachstum.

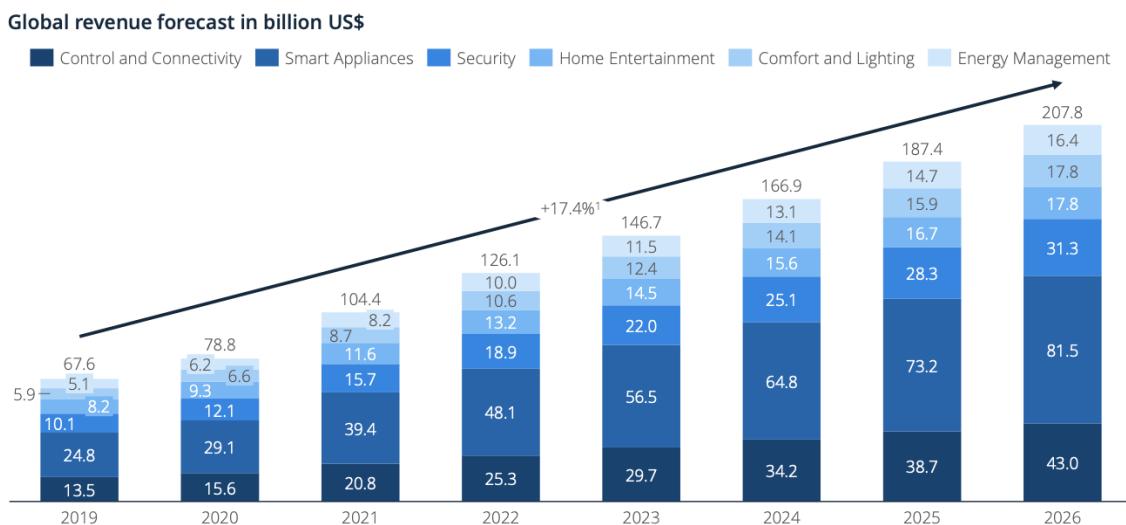
Der Abbildung (4.2) ist die nahezu Verdoppelung des Umsatzes bis 2026 zu entnehmen. Die Darstellung (4.3) der einzelnen Segmente zeigt die Zunahme am Weltmarkt von 2021 bis 2026 um ca. 100 Prozent. Die Zeitspanne von 2019 bis 2026 stellt einen durchschnittlichen Zuwachs von 17,4 Prozent dar. Anhand der Prognose und des Berichts von Statista ist deutlich zu sehen, dass der Smart Home Markt in den nächsten Jahren erheblich wachsen wird. Prognostiziert ist ein globaler Marktwert bei ca. 207,8 Milliarden US-Dollar bis 2026.



Source: Statista

statista

Abbildung 4.2: Umsatz-Prognose von Deutschland und den USA [LASQUETY-REYES 2021]



6 1:CAGR: Compound Annual Growth Rate / average growth rate per year
Sources: Statista Digital Market Outlook 2020

Abbildung 4.3: Globaler Smart Home Marktwert [LASQUETY-REYES 2021]

Schlüsseltechnologien und Barrieren

Unter den Schlüsseltechnologien im Smart Home sind Komponenten zu verstehen, die den Gedanken eines intelligenten Wohnraumes und Gebäudes forcieren. Dazu zählen unter anderem die Spracherkennung, die bei Sprachassistenten, darunter bspw. Amazon Alexa, Apple HomePod (Siri) und dem Google Nest, eingesetzt wird, sowie Artificial Intelligence (AI) und Künstliche Intelligenz (KI) zur Analyse, Auswertung und Optimierung von Verhaltensmustern und weiteren Analysezwecken [LASQUETY-REYES 2021]. Zu berücksichtigen sind dabei jedoch die dafür geeigneten Anwendungsfälle und die Akzeptanz der Nutzer ihre Verhaltensmuster analysieren zu lassen.

Interoperabilität

Die Kommunikation von Smart Home Geräten findet meistens über drahtlose Netzwerke auf Frequenz-Bandbreiten statt, die oft nicht miteinander kompatibel sind. Neben den drahtlosen Systemen gibt es auch, wie in Abschnitt (2.3.1) beschrieben, weitere Übertragungsmethoden, darunter Strom- und Datenleitungen. Ursächlich für die Inkompatibilität der Übertragungsmethoden ist die Entwicklung eines Protokolls für einen bestimmten Zweck, der darauf abgestimmt ist, den Anwendungsfall abzudecken und eine Markteintrittsbarriere zu schaffen, um einen Wechsel zwischen Anbietern zu erschweren [LASQUETY-REYES 2021]. Die damit einhergehende Schwachstelle eines Smart Home ist, dass die Geräte mit den bereits entwickelten Protokollen genutzt werden. So ist die Kommunikation über verschiedene Methoden nicht vorgesehen, was eine fehlende Interoperabilität zur Folge hat. Dies würde die Kombination von Geräten erschweren oder gar verhindern, die eventuell für den Anwender wünschenswert wäre. Um dies zu ermöglichen, müssten Integrationen oder Plug-ins von bestehende Plattformen zur Verfügung gestellt werden oder allgemein kompatible Kommunikationsprotokolle entwickelt werden, die bspw. mehrere Protokolle kombinieren können, darunter z.B. Zigbee2MQTT. Voraussetzung dafür ist die Fähigkeit zur Kompatibilität und die Übereinstimmung der Technologie, auf die das Protokoll aufbaut. Der Anwender muss vorab selbst prüfen, welche Geräte miteinander kompatibel sind [LASQUETY-REYES 2021].

Die Einführung von Bluetooth Low Energy (LE) Mesh¹ ist eine aktuelle Entwicklung, um der Herausforderung der Bewältigung der Interoperabilität einen Schritt näher zu kommen. Bei cloudbasierten Sprachdiensten muss ebenso die Kompatibilität geprüft werden. Neben der Datensicherheit und dem Schutz der Privatsphäre ist die fehlende Interoperabilität ein weiterer kritisch zu betrachtender Aspekt von SH Lösungen.

Derzeit häufig verwendete Protokolle sind unter anderem Bluetooth, Wireless Local Area Network (WLAN) (WiFi), KNX, ZigBee, Z-Wave, MQTT und weitere (siehe 2.4). Um Beispielsweise mit ZigBee über MQTT kommunizieren zu können, gibt es ein Framework, welches die Interoperabilität der beiden Protokolle ermöglicht. Dies ist das sogenannte *Zigbee2MQTT* Framework².

¹Bluetooth Mesh Technologie. <https://www.bluetooth.com> Besucht am 23.05.2022

²Erschafft eine Brücke zwischen ZigBee und MQTT. <https://www.zigbee2mqtt.io/> Besucht am 23.05.2022

4.2 Zielgruppenanalyse

Um den Adressaten dieser Arbeit näher zu definieren, erfolgt in diesem Abschnitt eine Zielgruppenanalyse. Im Rahmen dieser Tätigkeit wird die konkrete Zielgruppe identifiziert, die das Framework als Grundlage nutzt, um die individuellen Anwendungsfälle in der Umgebung des Büros umzusetzen und ggf. zu erweitern. Die Verwendung des Frameworks ist nicht zwangsläufig auf die Büroumgebung beschränkt, ebenso kann das Framework in privatem Gebrauch angewendet werden. Da es sich bei der Zielgruppenanalyse um einen fortlaufenden Prozess handelt, wird in dieser Arbeit die erste Iteration durchgeführt, die anschließend weiter verfolgt werden kann.

4.2.1 Ziel der Zielgruppenanalyse

Die Absicht einer Zielgruppenanalyse³ ist die Identifizierung der Personengruppen, die als potentielle Nutzer eines Produktes oder eines Marktsegmentes gelten. Diese Methodik ist ein relevantes Werkzeug in der Produktkonzeption und -entwicklung als auch in der Marktforschung. Maßnahmen und Anforderungen können aus der Zielgruppenanalyse abgeleitet und erarbeitet werden. Ein weiteres Ziel ist das bessere Kennenlernen der Zielgruppe, um dadurch deren Bedürfnisse und Interessen genauer zu identifizieren und zu betrachten.

4.2.2 Zielgruppendefinition

Zur Bestimmung einer Zielgruppe werden folgende Kriterien unter Verwendung des Prozesstemplates der *Target Group Analysis* verwendet:

- Beruf: *Ist das System nur für bestimmte Berufsgruppen interessant?* - Im Kontext der Arbeit, in der sich auf das smarte Büro bezogen wird, stehen Personen mit Programmierkenntnissen im Vordergrund.
- Bildung: *Über welchen Bildungsstand verfügt die Zielgruppe, die das System anspricht?* - Überwiegend Personen mit Ausbildung im IT-Umfeld sowie Akademiker im Bereich der Softwareentwicklung sind bedeutsam.
- Interessen: *Für Personen welcher Interessengebiete ist das System relevant?* - Personen die Innovationen vorantreiben, aktuelle und neueste Technologien und Frameworks im Bereich SH nutzen, könnten sich für dieses Konzept begeistern.
- Werte: *Über welche Werte verfügen die Interessenten?* - Von Bedeutung könnten die folgenden Werte sein, wie Weiterentwicklung, Ordnung, Disziplin und Selbstbestimmung, da das Framework ein selbstständiges Erarbeiten von Lösungen für Anwendungsfälle bietet.

Mithilfe der Zielgruppendefinition kann die Bestimmung und Eingrenzung der Interessenten erfolgen. Diese wird in folgendem Abschnitt verwendet, um die Zielgruppe zu erläutern. Stützend dazu werden bereits erhobenen Daten durch offizielle Statistiken und Umfragen genutzt.

³Beschreibung und Definition der Zielgruppenanalyse. <https://www.ecology.net/magazine/target-group-analysis> Besucht am 24.05.2022.

4.2.3 Zielgruppe

Die in der Marktanalyse (4.1) identifizierten Segmente werden in der Abbildung (4.4) nochmals aufgegriffen. Hierbei wird in der repräsentativen Statistik und Prognose der Statista GmbH die Nutzung des jeweiligen Segments veranschaulicht. Der Fokus dieser Prognose liegt auf der verstärkten Vertretung eines Segments in einem Smart Home in Deutschland. Die derzeit am meisten eingesetzten Segmente sind *Vernetzung und Steuerung* (rot) und *Komfort und Licht* (gelb). Das am wenigsten genutzt Segment stellt die *Gebäudesicherheit* (schwarz) in der Prognose dar. Im Jahr 2021 lag die Nutzung von Geräten des Segments *Vernetzung und Steuerung* bei 6,6 Millionen Nutzern, dicht gefolgt von dem Segment *Komfort und Licht* mit 6,5 Millionen Nutzern. Dagegen liegt das Segment *Home Entertainment* im Jahr 2021 bei 3,8 Millionen Nutzern.

Die bis 2026 veröffentlichte Prognose sagt voraus, dass die jeweiligen Segmente stark zunehmen und sich jeweils vervierfachen.

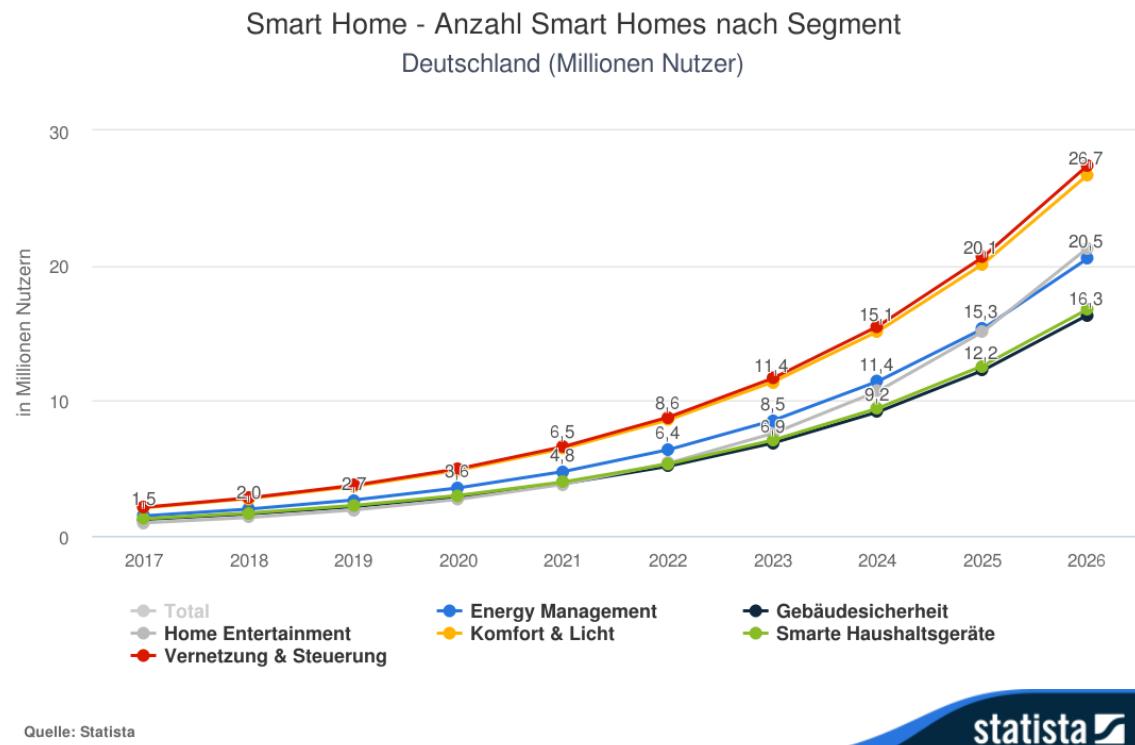


Abbildung 4.4: Anzahl Smart Home nach Segment [LASQUETY-REYES 2021]

Das Schaubild zeigt die prognostizierte Anzahl an Personen, die bereits Komponenten der Segmente *Vernetzung und Steuerung* und *Komfort und Licht* nutzen. Daraus könnte man die Hypothese entwickeln, dass sich dieses Wachstum ebenso auf Unternehmen übertragen lässt.

Im Fokus der Arbeit steht der Softwareentwickler als Anwender, welcher das System betreibt

und auf die individuellen Bedürfnisse anpasst.

Die Zielgruppe, bzw. Anwender des Frameworks für ein smartes Büro sollten gewisse Anforderungen mitbringen. Es ist eine grundlegende Kenntnis der Programmierung erforderlich. Im Rahmen dieser Arbeit handelt es sich um die Programmiersprache Java. Einer Umfrage der Developer Nation⁴ zufolge, benutzen von ca. 12 500 Befragten im Jahr 2021 35.8 Prozent die Programmiersprache Java. Somit kann ein Rückschluss erfolgen, dass Java eine der am häufigsten verwendete Programmiersprache ist.

Laut des Berichts der Developer Nation⁵ des dritten Quartals 2021 gibt es zu diesem Zeitpunkt 26.8 Millionen Softwareentwickler weltweit. Davon sind dem Bericht von Daxx⁶ zufolge 0.9 Millionen in Deutschland angesiedelt.

Eine vergleichbare Anwendergruppe ist die der Softwarelösung openHAB. Diese ist weltweit bekannt und zählt zu den beliebtesten und am meist genutzten Open-Source-Lösungen. Anhand den eigenen Statistiken kann die Community⁷ der openHAB Foundation ca. 42 000 Anwender vorweisen. Dem GitHub Repository der *openhab-core* Software sind 73 Mitwirkende zugeschrieben, die dazu beitragen die Software weiterzuentwickeln, bzw. zu verbessern und zu stabilisieren.

Damit die adressierte Zielgruppe etwas greifbarer wird und die Anwendungsschritte der *target group analysis* und dem *user-centered design* eingehalten werden, sind Persona⁸ entwickelt worden. Diese geben die Anwenderzielgruppe wieder und vermittelt einen Eindruck von Personen, die sich mit dem Softwareprodukt, bzw. mit dem Konzept auseinandersetzen. Ein konkretes Beispiel ist folgender Abbildung zu entnehmen, weitere Exemplare sind dem Anhang beigefügt (D).

⁴Umfragen, Statistiken rundum Softwareentwickler. <https://www.developernation.net/developer-reports/dn21> Besucht am 27.05.2022

⁵Berichte rundum Softwareentwickler. <https://www.developernation.net/developer-reports/dn21> Besucht am 27.05.2022

⁶Zusammenfassung mehrerer Berichte. <https://www.daxx.com/de/blog/entwicklungstrends/anzahl-an-softwareentwicklern-deutschland-weltweit-usa> Besucht am 27.05.2022

⁷Website Statistik der openHAB Community. <https://community.openhab.org/about> Besucht am 27.05.2022

⁸Eine repräsentative Vorstellung einer Person. <https://www.romanpichler.com/the-persona-template/> Besucht am 28.05.2022

ROMAN'S PERSONA TEMPLATE

 romanpichler

 PICTURE & NAME	 DETAILS	 GOAL
<p>What does the persona look like? What is its name? Choose a realistic and believable picture and name.</p> <p>Photo by ThisEngineeredRAE on Unsplash</p>  <p>Jonathan Maier</p>	<p>What are the persona's relevant characteristics and behaviours? For instance, demographics, such as age, gender, occupation, and income; psychographics, including lifestyle, social class, and personality; and behavioural attributes like usage patterns, attitudes, and brand loyalty. Only list relevant details.</p> <p>Geschlecht: Männlich Alter: 36 Beruf: Software Engineer Ausbildung: Diplom Informatiker/in (FH) Familienstand: Ledig, zwei Kinder (männlich im Alter von 7 Jahren) Hobbies: Klettern, Rad fahren Attribute: modern, karriereorientiert, nutzt keine soziale Medien, innovative Ideen, lernt gerne neues, gerne unter Leuten, strukturiert Lebensstil: Typ A (kulturell interessiert, modern, lebt den digitalen Lebensstil), gestaltet den Alltag und Lebensraum nach seinen Vorstellungen Persönlichkeit: eigenständig, zielstrebig, erfinderisch, flexibel, neugierig und innovativ Nutzungsmuster: Exklusiv, legt Wert auf Qualität, Performance und Digitalisierung</p>	<p>What problem does the persona want to solve or which benefit does the character seek? Why would the persona want to use or buy the product?</p> <ul style="list-style-type: none"> - Er möchte strukturierte Prozessabläufe, die wiederholbar sind - Das Wohlbefinden im Büro erhöhen - Büroaktivitäten erleichtern - Innovationen mit dem Büro kombinieren - Alle SmartHome Prozesse zentralisieren und möglichst schnell neue Funktionen und Use Cases abdecken - Mit moderner Technik in Berührung kommen - Möglichst viele Prozesse automatisieren

www.romanpichler.com
Template version 02/20

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 Unported license



Abbildung 4.5: Persona zur Zielgruppenanalyse

Der Zielgruppe werden folgende Eigenschaften, wie IT-affin, innovativ, zielstrebig, selbständige, flexibel, neugierig, erfinderisch, fortschrittlich beispielhaft zugeschrieben. Diese garantieren das Einsetzen und ständige Vorantreiben zum eigenen Nutzen.

4.3 Anwendungsfälle - Use Cases

Für die Erhebung und Ausarbeitung von Anforderungen an das zu entwickelnde Framework der Steuerzentrale und dessen einfache Handhabung der formalisierten Interaktionen des Softwareentwicklers wurden Anwendungsfälle, sogenannte *Use Cases* definiert. Diese helfen dabei, alle Ansprüche und Kriterien zu identifizieren und für das Konzept zu berücksichtigen. Im Rahmen des Requirements Engineering wurden die Anwendungsfälle dokumentiert und für die Konzeption, sowie für die Umsetzung des Prototyps, dem *Proof of Concept (PoC)*, einbezogen.

Das RE nach [POHL und RUPP 2021] gibt Schablonen, Vorgehensmodelle und Aufgaben vor, die

dabei helfen, den Kontext des Projektes zu erläutern und aus den anliegenden Sachverhalten und Anwendungsfällen die Anforderungen abzuleiten. Nach der strukturierten RE Methode Task-oriented Requirements Engineering (TORE) [ADAM, RIEGEL und DOERR 2014] werden die Aufgaben, die Systemfunktionen und die Interaktionen deutlich. Zum besseren Verständnis des Kontextes und der daraus resultierenden Anforderungen werden die jeweils generierten Anwendungsfälle in den folgenden Abschnitten erläutert.

4.3.1 Check-in mit einem Service-Roboter

Der erste Anwendungsfall skizziert den Check-in mit einem Service-Roboter. Daraus können Anforderungen abgeleitet werden, die für das Konzept eine wichtige Rolle spielen. Mithilfe der Erläuterung des *Use Cases* kann der Prozess aus mehreren Sichtweisen betrachtet werden, die zur Erhebung weiterer Anforderungen beitragen können.

Anhand dessen können die Schritte identifiziert werden, die notwendig sind, um die Interaktionen für den Softwareentwickler zu identifizieren. und so zu verallgemeinern, dass die formalisierten Interaktionen einfach zu handhaben sind. Des weiteren soll durch die Komplexität der Anwendungsfälle auch ersichtlich werden, welche Konzeptentscheidungen notwendig sind, damit über die Steuerzentrale Anwendungsfälle mit einem Service-Roboter abgebildet werden können.

Die Begrüßung und der Check-in eines Mitarbeiters oder angemeldeten Gastes an der Tür des Bürogebäudes erfordert mehrere Teilnehmer. Darunter zählt die jeweilige Person, die den Check-in erfährt, eine Kamera, mit der die davor stehende Person authentifiziert wird, ein Service-Roboter, der die Begrüßung und den Check-in durchführt und die Steuerzentrale, die den gesamten Prozess koordiniert.

Für den Zeitpunkt des Check-ins muss die Person an der Tür bekannt sein. Hierfür findet eine Authentifizierung der Person über die Kamera an der Eingangstür statt. Der Authentifizierungsvorgang wird bereits als gegeben vorausgesetzt und ist kein direkter Bestandteil der Steuerzentrale.

Sobald eine Person über die Kamera verifiziert wurde, wird eine MQTT-Nachricht über ein dafür definiertes Topic an den MQTT-Broker veröffentlicht, auf das die Steuerzentrale lauscht und dessen Inhalt konsumiert. Das Konsumieren des MQTT-Tops und die damit gesendete Nachricht wird als eingehendes Ereignis verwendet, worauf die Steuerzentrale reagiert. Basierend auf dem eingehenden Event wird der weitere Prozess gestartet. Nach dem Erhalt des Topics wird auf dessen Grundlage die Nachricht zugeordnet und die dafür vorgesehene Regel ausgeführt. Hierfür wird der Service-Roboter über die Steuerzentrale an die Tür geschickt, an der die Begrüßung stattfinden soll. Nach erfolgreichem Vorgang wird der Check-in durchgeführt, worauf die Person einen Platz buchen oder bei vorab getätigter Buchung diese bestätigen und sich als anwesend eintragen kann. Die grobe Skizzierung ist folgendem Diagramm zu entnehmen:

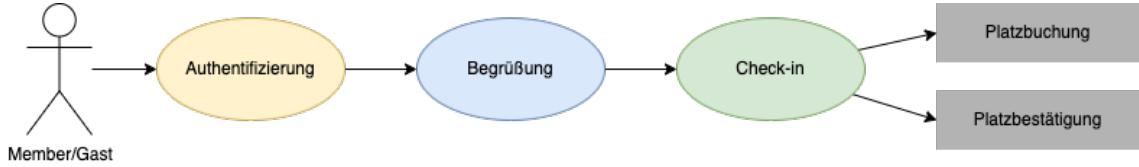


Abbildung 4.6: Use Case 1 - Anwendungsfalldiagramm

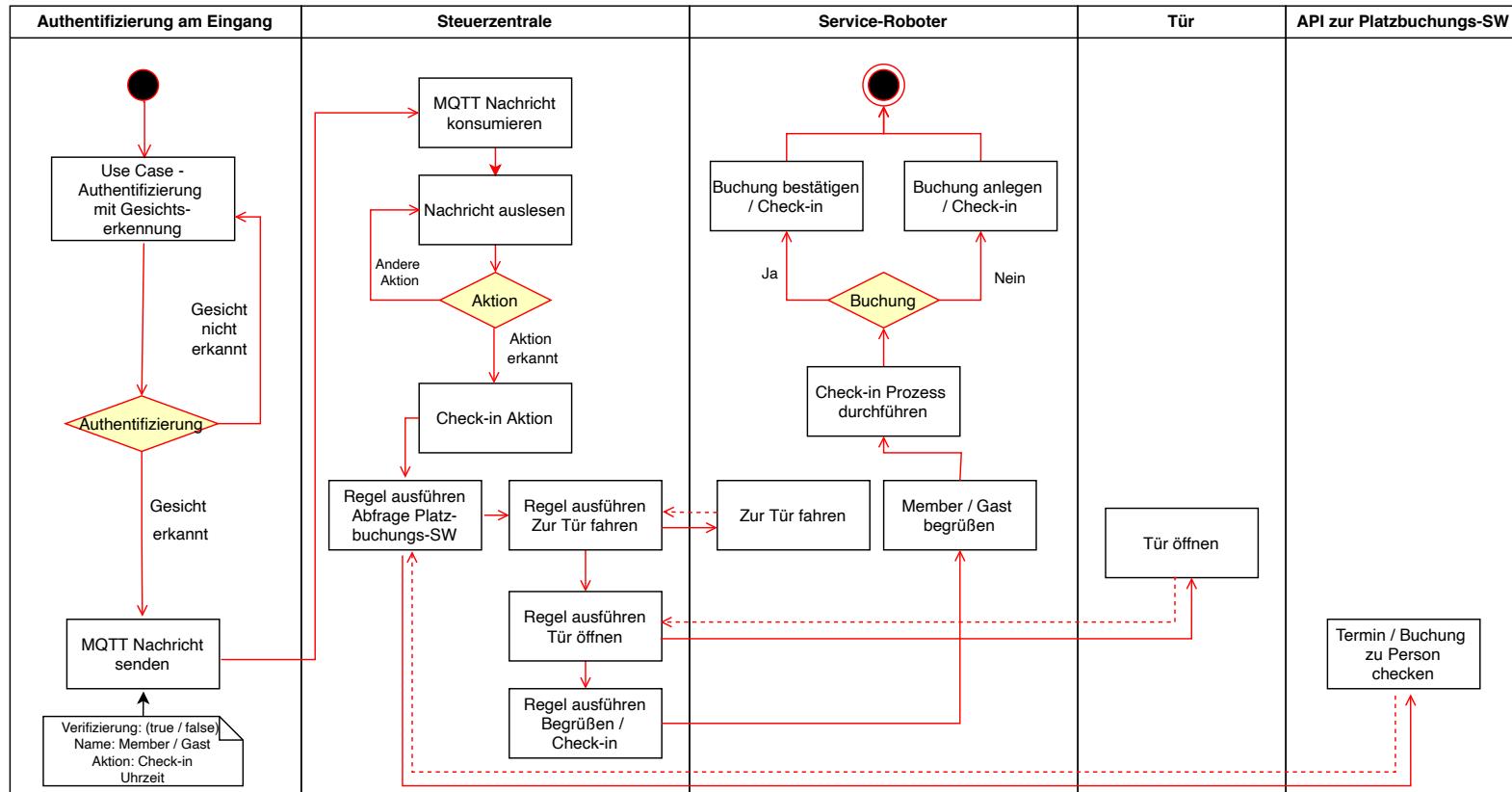
Sobald eine Person an der Tür authentifiziert und der Begrüßungsprozess gestartet wurde, werden über eine Schnittstelle zur internen Büroplatzbuchungssoftware die Platzbuchungen abgefragt. Währenddessen bewegt sich der Service-Roboter an die Tür, die sich daraufhin öffnet und die Person eintreten lässt. Zu aktuellen Zeitpunkt wird vorausgesetzt, dass die Person, die vor dem Service-Roboter steht auch diejenige ist, die an der Tür authentifiziert wurde. In zukünftiger Arbeit könnte die Person auch über die Kamera des Roboters erneut authentifiziert werden, um eine Übereinstimmung zwischen der Person an der Tür und der Person vor dem Service-Roboter zu erzielen. Damit wäre der korrekte Adressat des Check-ins gewährleistet. Dies ist für diesen Anwendungsfall jedoch nicht erforderlich und wird deshalb an der Stelle nicht weiter verfolgt.

Nachdem der Roboter die Person erkannt hat, startet die formale Begrüßung. Anschließend wird die Information der vorherigen Büroplatzbuchung verwendet, um das Einchecken der Person zu starten. Wurde bei der Abfrage der Platzbuchungen ein Eintrag der jeweiligen Person gefunden, so kann sich diese als anwesend einchecken und an den Arbeitsplatz gehen. Ist jedoch keine Buchung gefunden worden, so kann ein freier Platz über den Roboter reserviert, bzw. gebucht werden. Die Buchung wird anschließend an die Steuerzentrale zurückgegeben und über diese in das Buchungsportal eingepflegt. Nachdem dieser Schritt abgearbeitet ist, endet der Anwendungsfall und der Service-Roboter wird an seine Ausgangsposition, bzw. an dessen Ladestation geschickt. Somit ist das Szenario abgearbeitet und der Roboter steht für weitere Aufgaben zur Verfügung.

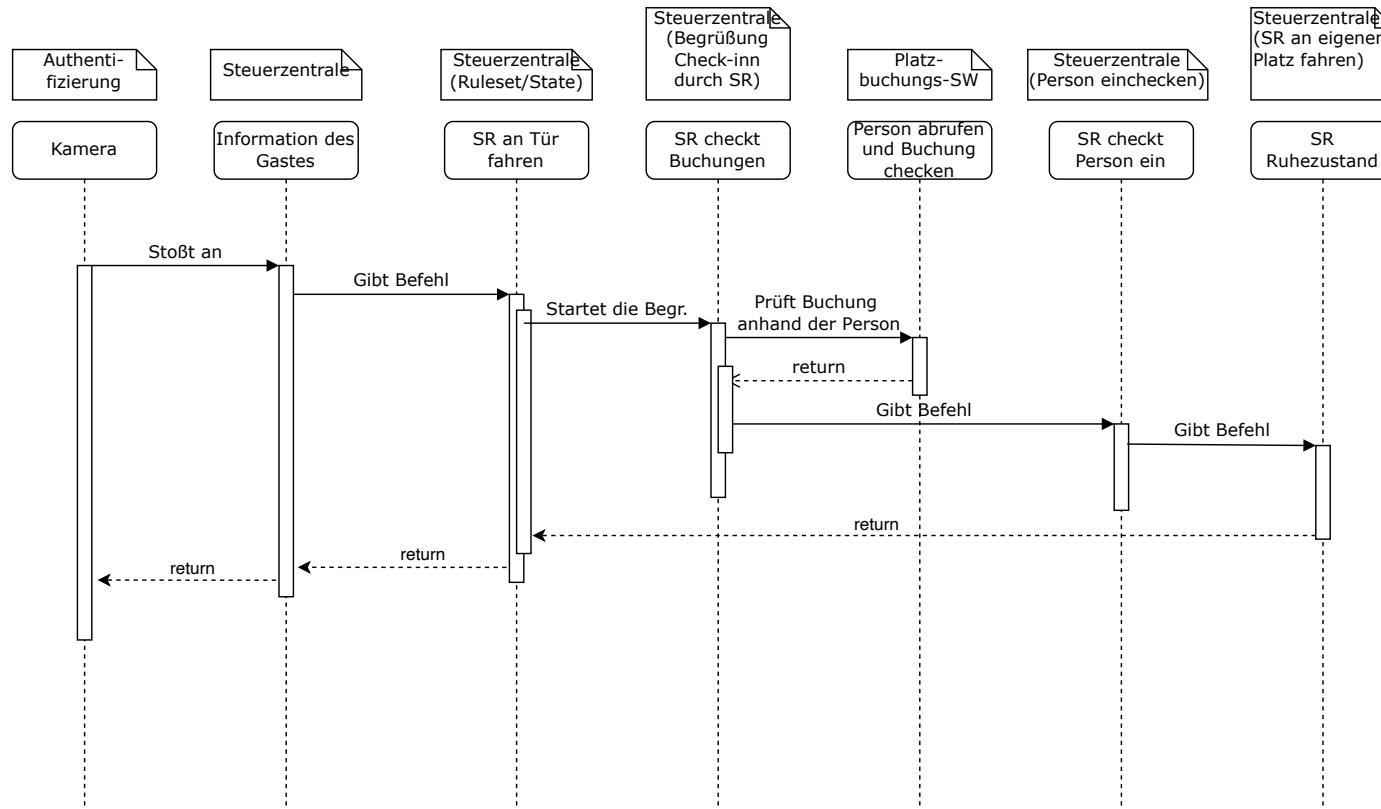
Eine konkrete Aufgabenbeschreibung, sowie eine definierte User Story und ein ergänzendes Ablaufdiagramm ist dem Anhang (siehe Anhang E) beigelegt.

Zur besseren Veranschaulichung des Anwendungsfalls sind die Prozesse visualisiert und grob skizziert. Den folgenden Diagrammen, darunter einem Aktivitätsdiagramm und einem Sequenzdiagramm, ist die visuelle Ergänzung zu entnehmen:

Use Case - Check-in mit einem Service-Roboter - Aktivitätsdiagramm



Use Case - Check-in mit einem Service-Roboter - Sequenzdiagramm



SR = Service-Roboter
SW = Software

4.3.2 Notfall-Evakuierung mit einem Service-Roboter

Mit dem Szenario der Notfall-Evakuierung wird ein weiterer Anwendungsfall definiert, der die Anforderungen zu identifizieren hilft, die für die Bereitstellung des Frameworks der Steuerzentrale benötigt werden.

Objektiv betrachtet gibt es bei diesem Fall ebenso einen Auslöser, bspw. einen Rauch- oder Gasmelder, der eine MQTT-Nachricht veröffentlicht, die über die Steuerzentrale konsumiert und verarbeitet wird. Die weiteren notwendigen Schritte, die der Entwickler vorab festgelegt hat, werden von dieser Steuerzentrale eingeleitet. Nachdem das MQTT-Topic, das die entsprechende Nachricht enthält, von der Steuerzentrale empfangen wurde, wird basierend auf dem Nachrichteninhalt die Regel und die darauffolgende Aktion angestoßen. Am Beispiel eines Rauchmelders als Auslöser würde die Steuerzentrale über die API Schnittstelle den vorab vom Entwickler festgelegten Check des internen Büroplatzbuchungssystems durchführen, um alle eingekenneten Personen und Platzbuchungen abzufragen und zwischenzuspeichern. Die Informationen werden genutzt, um den Service-Roboter an die Arbeitsplätze der jeweilig eingekenneten Personen zu schicken und über den Notfall zu informieren, bzw. zum Verlassen des Gebäudes aufzufordern. Ist eine Person nicht an ihrem Arbeitsplatz, soll der Service-Roboter den nächsten Arbeitsplatz ansteuern. Nach Kontrolle aller Arbeitsplätze soll der Service-Roboter die restliche erreichbare Bürofläche abfahren und nach Personen Ausschau halten. Die Erkennung der Person wird durch die Kamera des Roboters durchgeführt. Abschließend soll der Roboter an eine zentrale Stelle im Büro fahren und ohne Unterbrechung eine Durchsage starten und diese solange wiederholen, bis eine Person den Vorgang manuell beendet. Mit der Beendigung der Dauerschleife ist das Szenario abgeschlossen und der Roboter kann an seine Ausgangsposition zurückgeführt werden, sofern dies umgebungsbedingt noch möglich ist. Die grobe Skizzierung ist folgendem Anwendungsfalldiagramm zu entnehmen:

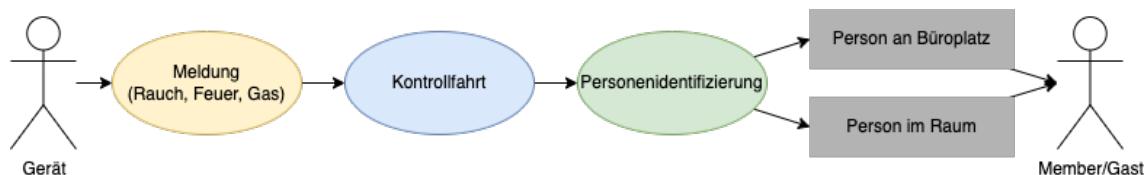
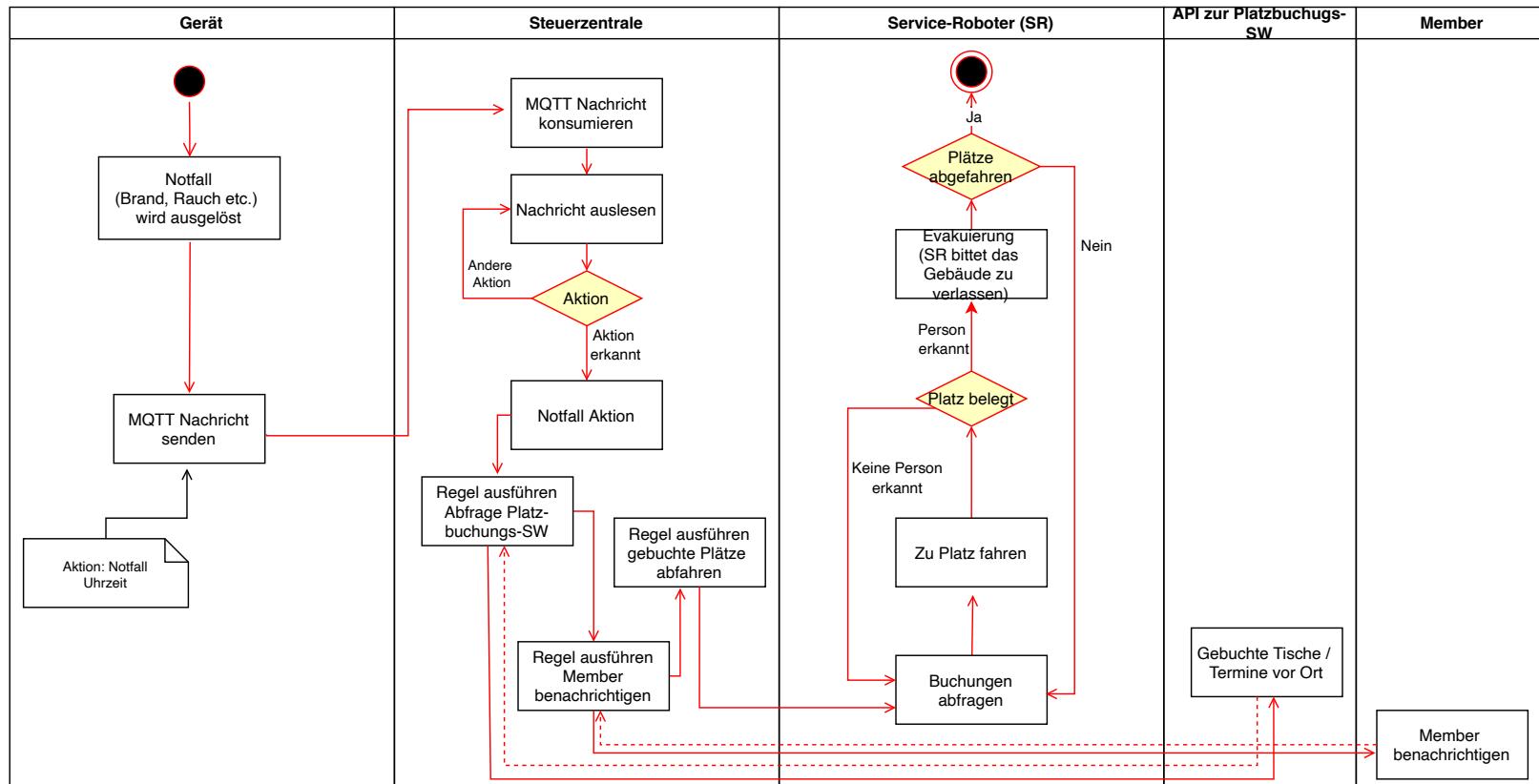


Abbildung 4.7: Use Case 2 - Anwendungsfalldiagramm

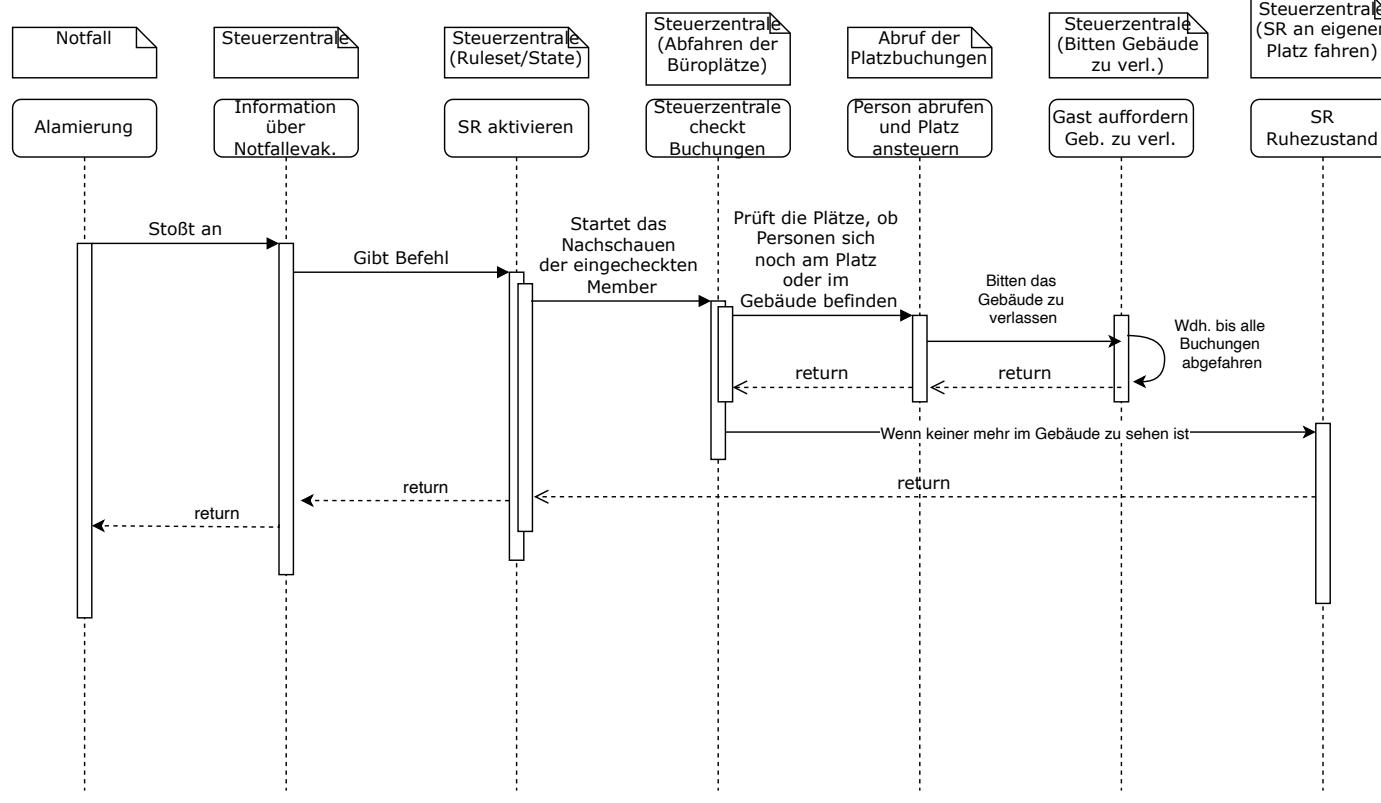
Die Anwendungsfälle wurden so gewählt, da diese eine gewisse Komplexität mit sich bringen, die es mit der Steuerzentrale abzudecken gilt.

Zur Stützung der textuellen Schilderung des zweiten Anwendungsfalls werden nachfolgend Diagramme dargestellt, die das Szenario widerspiegeln. Im Rahmen des RE wurden hierfür ebenso eine konkrete Aufgabenbeschreibung, sowie eine User Story und ein Ablaufdiagramm definiert. Diese sind dem Anhang (siehe Anhang E) zu entnehmen. Die folgenden Abbildungen setzen sich aus einem Aktivitätsdiagramm und einem Sequenzdiagramm zusammen:

Use Case - Notfall-Evakuierung mit dem Service-Roboter - Aktivitätsdiagramm



Use Case - Notfall-Evakuierung mit einem Service-Roboter - Sequenzdiagramm



SR = Service-Roboter

4.4 Experteninterview

Zur Analyse und Erhebung von Anforderungen, die sich an das System richten, werden Experteninterviews durchgeführt. Dabei wird sich, wie in dem Abschnitt (1.4.1) beschrieben, an dem unstrukturierten Ansatz der Führung eines solchen Interviews orientiert [ROBSON 2002]. Deshalb werden keine konkreten offenen oder geschlossenen Fragen gestellt. Der Aufbau des Interviews wird in weiteren Schritten aufgegriffen.

Die Ergebnisse der Interviews sind nicht repräsentativ, da sie im Rahmen dieser Arbeit mit einigen wenigen Experten durchgeführt wurden. Sie dienen lediglich der Informationsgewinnung und dem Einholen mehrerer Meinungsbilder, um aus allen Ideen, Gedankenanstößen und Ansichten ein Gesamtbild zu erzeugen und daraus mehrere Anforderungen zu gewinnen und abzuleiten.

4.4.1 Ziel des Experteninterviews

Ziel des Experteninterviews ist die Informationsgewinnung aus bestimmten Sachverhalten, für die es noch keine repräsentativen Umfragen gibt. Der Experte kann seine Sicht auf den Sachverhalt wiedergeben und neue Blickwinkel eröffnen. Dadurch können Rückschlüsse gezogen und Erfahrungen ausgetauscht werden. Diese sind oft wichtige Informationen zur Erhebung von Anforderungen zu einem bestimmten Sachverhalt.

4.4.2 Aufbau des Experteninterviews

Die Experteninterviews wurden im Gesamten als unstrukturierte Interviews durchgeführt. Lediglich die Rahmenbedingungen sowie der Einstieg in das Interview waren vorgegeben und ähneln sich bei jeder interviewten Person. Zu Anfang des Gesprächs wurde der Kontext und die Intension erläutert, damit der Experte die Situation und die Absichten kennenternt und die eigentliche Herausforderung erkennt. Grundlage dafür war die Erläuterung der Zielsetzung der Arbeit (1.3) zur Verdeutlichung der Intension. Mit den identifizierten Anwendungsfällen (4.3), die als Basis zur Extraktion von Anforderungen und als potentiell umsetzbare Funktionalitäten gelten, wurden Szenarien veranschaulicht, die dabei halfen, das Anwendungsumfeld zu konkretisieren. Nach der Schilderung des Kontextes und der zugrundeliegenden Ausgangspunkte wurde das Gespräch in Richtung Anforderungen gelenkt. Hierbei lag der Fokus auf dem Sammeln von Ideen, Sichtweisen und Meinungen, die als Grundlage dienten oder gar direkte Anforderungen an das System ergaben. Dabei war der Ausgang des Gesprächs offen. Falls während eines Gespräches der Fokus verloren ging, bzw. Exkurse ein zu großes Ausmaß annahmen, wurde wieder auf die vorliegende Sachlage aufmerksam gemacht und der Fokus erneut auf die Anforderungen geworfen. Schwerpunkte bei den Interviews waren zum einen die Anforderungen, welche für eine einfache Handhabung der formalisierten Interaktionen für den Softwareentwickler gelten, und zum anderen die Funktionalitäten, die der Experte dem System gegenüber sieht, um ein Regelwerk für ein intelligentes Büro zu erstellen. Ebenso wurden nicht funktionale Anforderungen, die ein solches System mit sich bringen soll, adressiert. Die zusammengefassten Anforderungen und die daraus abgeleiteten Bedingungen sind dem Abschnitt (4.5) zu entnehmen.

4.4.3 Zusammenfassung der Experteninterviews

In Summe wurden insgesamt fünf Experteninterviews durchgeführt. Jedes Gespräch war individuell und hatte dementsprechend einen anderen Verlauf, bzw. ein anderes Ergebnis. Dennoch konnte der inhaltliche Fokus gewahrt und verschiedene Meinungsbilder eingeholt werden. Jeder befragte Experte konnte zu dem anliegenden Sachverhalt seine Meinung äußern und wichtige Informationen und Sichtweisen mitteilen. Die erhobenen Informationen wurden analysiert, aufbereitet und als Anforderungen dokumentiert. Die dabei entstandenen Informationen und Anforderungen werden in folgendem Abschnitt aufgezeigt.

4.5 Anforderungen

In Folge der vorangestellten Literaturrecherche, der Markt- und Zielgruppenanalyse, der entwickelten Anwendungsfälle und der durchgeföhrten Experteninterviews ergaben sich Anforderungen an das System, die in der Konzeption und der anschließenden prototypischen Implementierung zu berücksichtigen sind. Zur Dokumentation der nicht funktionalen Anforderungen wird sich an dem *Software Produkt Qualitätsmodell* nach der ISO Norm 25010⁹ orientiert.

Der folgenden Tabelle (4.1) sind die funktionalen Anforderungen (FA) zu entnehmen:

	Anforderung (User Story)	Priorität	Quelle
FA1	Als Softwareentwickler möchte ich mit einer vorgegebenen Struktur Regeln definieren, die zur Laufzeit ausgeführt werden können.	Essentiell	Zielsetzung der Arbeit
FA2	Als Softwareentwickler möchte ich, dass alle Regeln, die ich definiert habe, zu bestimmtem Auslöser gestartet werden.	Essentiell	Experteneinterview
FA3	Als Softwareentwickler möchte ich einen Zustandsraum erstellen, der individuell implementiert werden kann.	Essentiell	Experteneinterview
FA4	Als Softwareentwickler möchte ich Bedingungen definieren und abfragen können, die basierend auf dem Ereignis die dazugehörige Regel ausführen.	Essentiell	Experteneinterview
FA5	Als Softwareentwickler möchte ich Komponenten anlegen können, die reelle Gegenstände und dessen Zustände abbilden.	Hoch	Experteneinterview
FA6	Als Softwareentwickler möchte ich das vorgegebene Framework individuell nutzen können, indem ich bei der Definition der Regeln und der Informationsbeschaffung nicht eingeschränkt bin.	Essentiell	Zielsetzung der Arbeit, Experteneinterview

Tabelle 4.1: Funktionale Anforderungen (FAs)

⁹Qualitätscharakteristiken nach ISO. <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>
Besucht am 24.06.2022.

	Anforderung	Priorität	Quelle
NFA1	Benutzerfreundlichkeit (BF): Der Anwender soll nach Implementierung, bzw. Definition einer Regel, diese mit max. 3 Schritten in das Regelwerk einfügen können.	Hoch	Experten-interview
NFA2	BF: Mehr als 90% der Anwender sollten bei erster Verwendung des Frameworks in der Lage sein, Regeln anzulegen und diese dem Regelwerk hinzuzufügen.	Essentiell	Zielsetzung der Arbeit, Experteninterview
NFA3	BF: Der Anwender soll die Möglichkeit haben, zu dem Zeitpunkt der Verwendung des Frameworks alle verfügbaren Funktionen nutzen und anwenden zu können.	Mittel	Experten-interview
NFA4	BF: Der Anwender braucht sich zu keinem Zeitpunkt um das Management, bzw. um den Ablauf und Durchführung von Regeln kümmern. Er muss nur sicherstellen, dass die Regeln valide sind und dem Kontext passende Bedingungen und Regelausführungen festlegt.	Essentiell	Experten-interview
NFA5	Zuverlässigkeit (Z): Als Anwender möchte ich, dass zu 100% die Regel ausgeführt wird, die der Auslöser ansteuert. (Bei der Ausführung soll immer die Regel gestartet werden, die durch den Auslöser ausgelöst wurde.)	Essentiell	Experten-interview
NFA6	Performanz (P): Ausgelöste Regeln, die beide voneinander unabhängige Ressourcen oder Attribute im Zustandsraum belegen, sollen parallel ausgeführt werden.	Hoch	Anwendungsfall, Experten-interview
NFA7	P: Regeln, die über Kommunikationsprotokolle ausgelöst werden, sollen unter einer Sekunde starten. Die Zeit der Durchführung ist abhängig von der Regel selbst.	Mittel	Anwendungsfall, Experten-interview
NFA8	Verfügbarkeit (V): Die Steuerzentrale muss eine Verfügbarkeit von 99,9% vorweisen. (Zum aktuellen Zeitpunkt ist diese Anforderung zu vernachlässigen.)	Niedrig	Experten-interview
NFA9	V: Die Steuerzentrale muss während der Arbeitszeiten zwischen 7-18 Uhr ohne Ausfälle verfügbar sein. (Zum aktuellen Zeitpunkt ist diese Anforderung zu vernachlässigen.)	Niedrig	Experten-interview
NFA10	Fehlertoleranz: Syntaktisch oder inhaltlich fehlerhafte Regeln sollen nicht zum Absturz der Steuerzentrale führen.	Mittel	Experten-interview
NFA11	Kontrollierbarkeit, Beobachtbarkeit: Prozesse und Zustände der Steuerzentrale müssen eingesehen werden können, bspw. durch Monitoring oder über Oberflächen. (Zum aktuellen Zeitpunkt zu vernachlässigen.)	Niedrig	Experten-interview

Tabelle 4.2: Nicht funktionale Anforderungen (NFAs)

Eine Auswahl an zusätzlichen Anforderungen, die ergänzend zu den bereits Bestehenden aus der Zielgruppenanalyse und den Anwendungsfällen entstanden sind und sowohl die funktionalen (4.1) als auch die nicht funktionalen Anforderungen (4.2) erweitern, ist der folgenden Tabelle zu entnehmen. Diese sind als zusätzliche Anforderungen (ZAF) gekennzeichnet:

	Anforderung	Priorität	Quelle
ZAF1	Zur Entwicklung des Frameworks soll Java als Programmiersprache verwendet werden.	Hoch	Experteninterview, Zielgruppenanalyse
ZAF2	Die Abbildung von Komponenten soll über einen Zustandsraum dargestellt werden.	Niedrig	Anwendungsfall, Experteninterview
ZAF3	Zur parallelen Ausführung von Regeln soll ein Thread Pool eingesetzt werden, der die Regeln jeweils als eigenen Thread laufen lässt.	Hoch	Anwendungsfall, Experteninterview
ZAF4	Regeln sollen über zeitbasierte oder MQTT basierte Auslöser gestartet werden, nachdem die dafür vorgesehene Bedingung zutrifft. (Im Rahmen der Anwendungsfälle wird MQTT als Kommunikationsprotokoll und Auslöser gewählt.)	Hoch	Anwendungsfall
ZAF5	Regeln und damit einhergehende Aktionen dürfen erst nach aktivieren eines bestimmten Auslösers (Triggers) ausgeführt werden.	Essentiell	Experteninterview
ZAF6	Der Zustandsraum muss zur Laufzeit zur Verfügung stehen	Hoch	Experteninterview
ZAF7	Die Kommunikation erfolgt überwiegend durch MQTT	Hoch	Anwendungsfall
ZAF8	Sicherheit: Die Kommunikation soll über einen MQTT Broker im eigenen Netzwerk erfolgen. Das System soll nach außen nicht erreichbar sein.	Hoch	Experteninterview
ZAF9	Bereitstellung eines Single Point of Contact (SPC) (Implementierung, Anpassung und Erweiterung der Regel und der Logik).	Hoch	Zielgruppenanalyse, Experteninterview

Tabelle 4.3: Zusätzliche Anforderungen

Alle relevanten Anforderungen, die mithilfe den genannten und durchgeführten Erhebungsmethoden eruiert wurden, fließen in die Konzeption des Frameworks mit ein. Diese bilden die Grundlage des Konzeptes.

Kapitel 5

Konzeption

In diesem Kapitel wird das erarbeitete Konzept dargelegt. Basierend auf den Anforderungen, die aus den Anwendungsfällen, den Experteninterviews und der Zielgruppenanalyse erhoben wurden, werden die daraus generierten Überlegungen und Entscheidungen transparent dargestellt. Durch die bereits erfolgte Anforderungsanalyse (siehe Kapitel 4) sind erste Schritte der Konzeption abgeschlossen. Zu Anfang des Kapitels wird das allgemeine Ziel eines Konzeptes sowie die konkrete Absicht hinter dieser Arbeit erläutert. Anschließend wird auf das Anwendungsumfeld des Systems (5.2) eingegangen, darunter die zusätzlichen Komponenten, die notwendig sind, um das Framework in einer dafür vorgesehenen Umgebung sinnvoll einzusetzen. Darauffolgend wird anhand der zugrundeliegenden Informationen und Anforderungen das Architekturkonzept (5.4) ausgeführt. Dabei wird auf verschiedene kontextabhängige Aspekte sowie auf die Sichtweisen des Anwenders und des Frameworks eingegangen.

5.1 Ziel der Konzeption

Das Ziel einer Konzeption ist die Veranschaulichung von abstrakten Ideen und Entwürfen. Hierbei werden aus den zugrundeliegenden Problemstellungen, Szenarien und Anforderungen Entwürfe und Lösungsmöglichkeiten erarbeitet und identifiziert. Diese helfen bei der Aufstellung von notwendigen Schritten und dienen als Grundlage zur Untermauerung und Darlegung von Entscheidungen. Somit wird Dritten der Kontext, die Domäne und das zu lösende Problem, bzw. die Lösung dargestellt.

Das Konzept dieser Thesis erarbeitet eine Lösung zur Implementierung einer Anwendung, die es ermöglicht, Automatisierungen, Regeln und Prozesse innerhalb eines Firmenbüros zu koordinieren. Der Fokus liegt dabei verstärkt auf der einfachen Nutzung des Frameworks für den Anwender und die uneingeschränkte und schnell umzusetzende Ausprägungsvielfalt von Regelprozessen. Dadurch kann der Anwender die vorgegebene Struktur nutzen, um individuelle Sachverhalte zu realisieren, die in seinem Umfeld abzudecken sind. Im Rahmen dieser Arbeit werden verstärkt Anwendungsfälle mit einem Service-Roboter umgesetzt. Hierfür wird der allgemeine Aufbau der Architektur skizziert und demonstriert, wie eine solche Lösung aussehen kann. Unter Berücksichtigung der Forschungsfrage (siehe Abschnitt 1.2) wird eine Möglichkeit offengelegt, mit der ein Softwareentwickler neue

Regeln entwickeln und dem System hinzufügen kann, ohne ein weiteres zu erlernendes Framework zu verwenden. Dabei sollen die notwendigen Schritte und Interaktionen formalisiert und für den Entwickler vereinfacht werden.

5.2 Anwendungsumfeld

Grundsätzlich ist der Einsatzort des Frameworks variabel, da die eigentliche Implementierung und Nutzung der Regeln und Prozesse abhängig vom Anwender und dessen Umfeld sind. Dadurch kann sowohl im privaten Smart Home Umfeld als auch in Büroräumen ein System mithilfe des Frameworks aufgebaut werden. Basierend auf den vorangestellten Tätigkeiten, darunter die Anforderungsanalyse, und der Eingrenzung auf den Einsatz im Smart Office liegt darauf der Fokus.

Stützend auf den vorab ermittelten Anwendungsfällen (siehe Abschnitt 4.3.1 und 4.3.2) sind unabhängige Komponenten, darunter bspw. ein Service-Roboter und weitere einsatzfähige Geräte, sowie ein MQTT-Broker notwendig. Ein MQTT-Broker wird im Zuge der Konzeption von dem Framework selbst nicht bereitgestellt, lediglich die Client-Anbindung wird gegeben. Der Anwender muss sich selbstständig um den verfügbaren Broker kümmern. Dadurch kann die Anforderung einer Kommunikation mittels MQTT umgesetzt werden. Eine denkbare infrastrukturelle Architektur kann wie folgt aussehen:

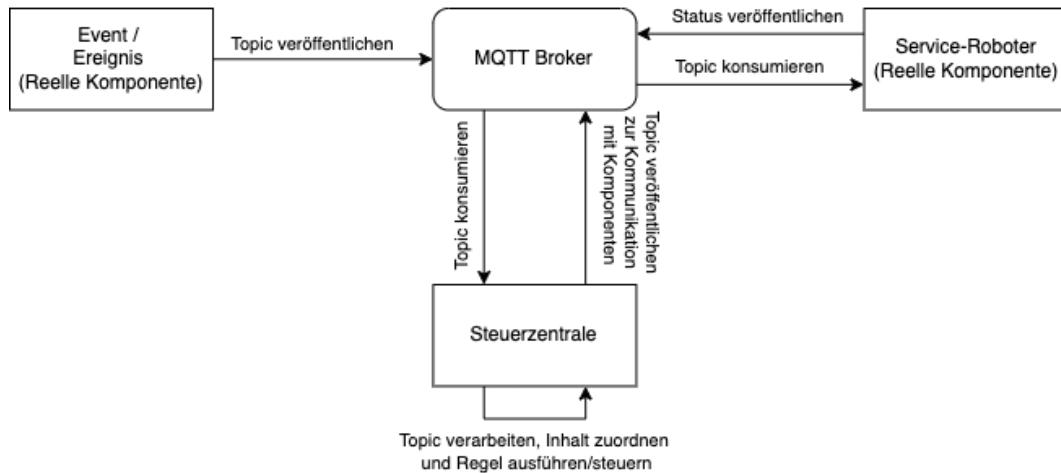


Abbildung 5.1: Infrastruktur des Anwendungsumfeldes der Steuerzentrale

5.3 Konzept

Die Intension, die hinter der Ausarbeitung dieses Konzeptes steht, ist zum einen die einfache Handhabung der formalisierten Interaktionen für Softwareentwickler unter der Verwendung des Frameworks und zum anderen die offene Gestaltung von Regelprozessen. Dadurch ist der Anwender bei der Implementierung von Regeln, Aufgaben und Automatisierungen für ein intelligentes Büro

nicht eingeschränkt und kann mit der Regeldefinition flexibel variieren. Es soll lediglich ein Muster vorgegeben werden, damit Regeln einheitlich als solche von dem Framework verarbeitet und genutzt werden können.

Bei vergleichbaren Softwareprodukten, die im Rahmen dieser Arbeit erläutert wurden (siehe Kapitel 2.4 und 2.5), ist die Vielfalt der Regelausprägung auf den Kontext des Systems eingeschränkt und benötigt mehrere Schritte, um sie zu realisieren. Dies bedeutet, dass Regeln und Prozesse nur mit Komponenten und Informationen innerhalb des Systems arbeiten können, bzw. benötigte Informationen erst durch eine systemseitige Erweiterung durch Plugins verfügbar sind, auf die der Nutzer keinen direkten Einfluss nehmen kann. Mit den bestehenden Lösungen wird versucht, die Regeldefinition für Endnutzer so einfach wie möglich zu gestalten, wodurch eventuell erst zahlreiche Selektionen über die Benutzeroberfläche zum gewünschten Ziel führen.

Um dennoch dem Anwender eine Struktur vorzugeben, mit der Regeln definiert und zur Laufzeit der Anwendung ausgeführt werden können, soll mit diesem Konzept ein Framework zum Lösen dieser Herausforderungen erarbeitet werden. Hierfür soll der Anwender mit der Komplexität der Regelverwaltung und deren Durchführung nicht konfrontiert werden. Dieser ist lediglich in der Verantwortung, die für ihn notwendigen Regeln und Prozesse zu definieren und dem Framework bereitzustellen. Dadurch soll dem Framework-Verwender die Möglichkeit geboten werden, das zur Verfügung gestellte System mit individuellen Regeln, dafür vorgesehenen Bedingungen, Komponenten und deren Zustände zu implementieren. Beispielsweise können bei Regeldefinitionen Informationen direkt von Datenpunkten über HTTP-Abfragen bezogen werden, bzw. Ressourcen und Inhalte individuell nachgezogen werden.

Das Konzept beinhaltet die in folgendem Abschnitt dargelegten grundsätzlichen Komponenten.

5.3.1 Konzeptkomponenten

Für die Erläuterung des Konzeptes werden vorab die einzelnen Konzeptkomponenten dargelegt, da diese die Anhaltspunkte für den weiteren Verlauf und den Kern des Frameworks darstellen:

- Regel (Rule): Eine Regel ist ein Konstrukt, welches bei bestimmten Events die darin enthaltenen Aktionen ausführen soll. Der grobe Aufbau einer Regel ist immer gleich. Diese beinhaltet einen Auslöser, eine Bedingung, einen Prozess und einen eindeutigen Namen. Die Inhalte der Regel kann der Anwender ja nach Bedarf beliebig ausprägen und ergänzen.
- Komponente (Component): Eine Komponente soll einen reellen Gegenstand abbilden, der bei Benutzung durch eine Regel unter anderem für weitere gesperrt und danach freigegeben werden kann. Diese Komponente kann ebenso sämtliche Zustände beinhalten und als Objekt in den Zustandsraum aufgenommen werden.
- Zustandsraum (State): Der Zustandsraum, das sog. Zustandsobjekt, soll alle Zustände von Komponenten und weiteren Geräten abbilden. Dieser repräsentiert die Zustände der Realität.
- Auslöser (Trigger): Ein Auslöser löst im Allgemeinen eine Zustandsänderung aus. Im Kontext des Frameworks gibt es zwei Ausprägungen von Auslösern. Zum einen werden eingehende Ak-

tionen entgegengenommen und über die Transformation zu einer Zustandsänderung verarbeitet, zum anderen können durch Regelprozesse wiederum weitere Schritte und Zustandsänderungen ausgelöst werden. Dadurch werden ausgehende Aktionen gesteuert.

- Transformation (Transformer): Mit der Transformation werden eingehenden Zustandsänderungen, die durch ein Event oder Auslöser ausgelöst werden, auf das eigentliche Zustandsobjekt übertragen.

Das Konzept wird aus zweierlei Sichtweisen betrachtet, die in folgendem Abschnitt aufgegriffen werden und für die weitere Konzepterläuterung notwendig sind.

5.3.2 Sichtweisen

Wie bereits aus dem Kontext hervorgeht, wird das System aus zwei Perspektiven beleuchtet. Zum einen aus Sicht des Anwenders, der das Aufsetzen und in Betrieb nehmen des Systems sowie das Definieren und Implementieren von Regeln als Aufgabe übernimmt. Zum anderen die Bereitstellungssicht, die das Framework als ausführende Kraft besetzt. Dieses sorgt für die Erfüllung der vom Anwender definierten Regeln. Ebenso stellt es Funktionen bereit, die das Empfangen von Events, das Starten von Regeln und Prozessen ermöglicht. Das Management dazu wird vom Framework übernommen. Das Konzept widmet sich grundlegend dem initialen Aufbau des Frameworks.

Der Anwender hat das initiale Setup des Frameworks zur Aufgabe. Zuerst sollte der Zustandsraum erstellt werden. Darin sind alle notwendigen Zustände als Attribute und Objektfelder abzubilden. Diese können konkret Geräte sein, die im Rahmen des intelligenten Büros verwendet werden, bspw. ein Service-Roboter, ein Türöffner und weitere steuerbare Geräte, die ihren Zustand ändern können. Nachdem das Zustandsobjekt definiert wurde, geht es in die Implementierung der Regeln. Diese sind individuell je nach Anforderungen des Anwenders zu erstellen. Jedoch sind bestimmte Funktionen und Vorgaben bezüglich Bedingungsprüfung und Regeldurchlauf einzuhalten. Anforderungen dabei sind das Implementieren von drei unabdingbaren Funktionen. Die erste Funktion ist die Zuordnung des Auslösers, der bei einem bestimmten Event eine Regel durchläuft. Die zweite zu implementierende Funktion ist die Prüfung von Werten im Zustandsraum, sodass beim Eintreten spezifischer Bedingungen und Zuständen bestimmte Regelprozesse ausgeführt werden können. Die dritte vorgegebene Funktion ist die des eigentlichen Regeldurchlaufs. Die darin enthaltenen Aufgaben werden nach zutreffender Bedingung abgearbeitet.

Nach Fertigstellung aller drei Funktionen muss das Objekt dem System übergeben werden. Anschließend ist die konkrete Transformation zu definieren. Darüber wird festgelegt, welches eingehende Event, bzw. MQTT-Topic eine bestimmte Zustandsänderung bewirkt. Erst nach Änderung des Zustandes wird die Überprüfung und Ausführung von Regeln ausgelöst. Das konkrete Konstrukt der Transformation wird im Kapitel der Umsetzung (siehe Abschnitt 6.2.4) erläutert. Abschließend sind die Kommunikationswege zu aktivieren. Da MQTT zum aktuellen Zeitpunkt und zur Abdeckung der Anforderungen die einzige Kommunikationsschnittstelle abbildet, ist die Konfiguration des MQTT-Clients einzustellen. Hierfür muss der Entwickler dem Framework den Host des MQTT-Brokers sowie den Nutzernamen des Clients und dessen Passwort übergeben. Über das Framework wird dann das Setup durchgeführt, sodass Topics (Themen), die über den Broker veröffentlicht

werden, konsumierbar sind. Im Zuge dessen muss der Anwender die Topics innerhalb der jeweiligen Transformation definieren, wodurch gewährleistet wird, dass die Steuerzentrale nur auf diese reagiert. Durch die Transformation erfolgt ebenso die Zuordnung, welches Zustandsattribut mittels eines bestimmten Topics geändert werden soll. Hierfür ein Beispiel:

Sobald eine Person an der Tür authentifiziert wurde, wird ein Thema mit dem Namen derjenigen Person übergeben. Basierend auf dem eingehenden Topic wird dann für dieses Ereignis der Wert im Zustandsraum auf den Namen der Person geändert. Somit kann die Steuerzentrale mit dem neuen Wert und der Änderung des Zustandsraumes arbeiten. Durch die Zustandsänderung wird dann geprüft, ob eine Regel für diese bestimmte Änderung definiert wurde. Trifft diese Bedingung zu, so wird der dazugehörige Regelprozess durchlaufen.

Sind diese Schritte abgearbeitet, sind die Pflichten des Anwenders erfüllt und die Steuerzentrale kann gestartet werden.

Im Nachfolgenden wird jetzt der Durchlauf vom Eingang einer MQTT-Nachricht bis zur Ausführung einer Regel aus Sicht des Frameworks erläutert.

Nach erfolgreichem Einstellen und Starten des Systems soll der MQTT-Client hochgefahren werden. Dieser baut daraufhin eine Verbindung zu dem MQTT-Broker auf und startet das Zuhören auf eingehende Themen. Dabei soll bereits nach den Topics, die der Nutzer über die Einstellungen im Rahmen der Transformation übergeben hat, gefiltert werden. Sobald eine der bekannten MQTT-Nachrichten konsumiert wurde, soll mittels des Transformers der Zustandsraum, bzw. das Zustandsobjekt auf die im Topic enthaltenen Informationen abgeändert werden. Der Transformer stellt den Ausgangspunkt für die Transformation des Nachrichteninhalts zur Zustandsänderung dar. Innerhalb dieser Komponente findet die Zuordnung des Topics zu den darauf adressierten Feldern des Zustandsobjektes statt. Mittels der Nachricht sollen die Informationen dieser in den Attributwert der Variable, die durch den Entwickler spezifiziert wurde, übertragen werden. Die Änderung des Zustandsraumes löst daraufhin den weiteren Ablauf aus.

Das aktuelle Zustandsobjekt wird zum Zeitpunkt der Überschreibung für weitere Lese- und Schreiboperationen durch einen Lock-Mechanismus gesperrt, sodass die Gültigkeit des Zustandsraumes nicht erlischt und nach wie vor reale Zuständen wiedergegeben werden. Anschließend erfolgt die Erstellung einer Kopie des Zustandsobjektes, mit der darauffolgend weiter verfahren wird. Nachdem das Zustandsobjekt geklont wurde, wird der Sperrvorgang aufgehoben und die Überprüfung aller Regeln, die dem Framework bekannt sind, iteriert, d.h. es werden alle Regeln der Liste durchlaufen bis eine Bedingung zutrifft. In diesem Prozess wird jeweils nochmals eine Kopie des Zustandsobjektes erzeugt, mit dem bei der Iteration die Regel-Bedingungen überprüft werden. Bei zutreffender Bedingung wird die entsprechende Regel ausgeführt. Zur Abarbeitung mehrerer Prozesse gleichzeitig soll durch Asynchronität die Bedingungsprüfung und Durchführung der Regel in einen separaten *Thread* ausgelagert werden, sodass Regeln parallel ausgeführt werden können. Wichtig an dieser Stelle zu erwähnen ist die Prüfung der Regel-Bedingungen mittels der Kopie des Zustandsraumes. Durch den Sperrvorgang wird gewährleistet, dass eine Zustandsänderung sauber durchgeführt wird und daraufhin eine Regel mit der Kopie des Zustandsobjektes ausgelöst werden kann. Hierdurch wird eine Zustandsänderung zu einem Zeitpunkt stattfinden und die dafür definierte Regel ausgeführt,

sofern die Bedingung im Vorfeld ebenso zutrifft.

Wird kurz darauf eine weitere Nachricht konsumiert, so kann das aktuelle Zustandsobjekt dahingehend überschrieben und erneut kopiert werden. Die Kopie wird wiederum für die Regel-Iteration verwendet und tangiert die vorherige Kopie nicht. Somit soll zu jedem Zeitpunkt der aktuelle Zustandsraum dargestellt werden.

Innerhalb eines laufenden Regelprozesses kann wiederum eine Zustandsänderung erfolgen, die daraufhin eine weitere Regel ausführen kann. Dies soll durch die inverse Funktion der Transformation geschehen, indem der Anwender in dem Regelprozess lediglich die Zustandsänderung beschreibt. Das Framework nutzt auf deren Grundlage die vorab durch die Transformation übergebenen Informationen, damit zum Zeitpunkt der Regelimplementierung dem Anwender nicht zwangsläufig das Topic für die Kommunikation über MQTT bekannt sein muss. Die Kenntnis des Topics und dem darauf zugewiesenen Feld des Zustandsraumes ist nur bei initialer Definition erforderlich.

Dem folgenden Diagramm ist der abstrakte Programmablauf des Frameworks zu entnehmen:

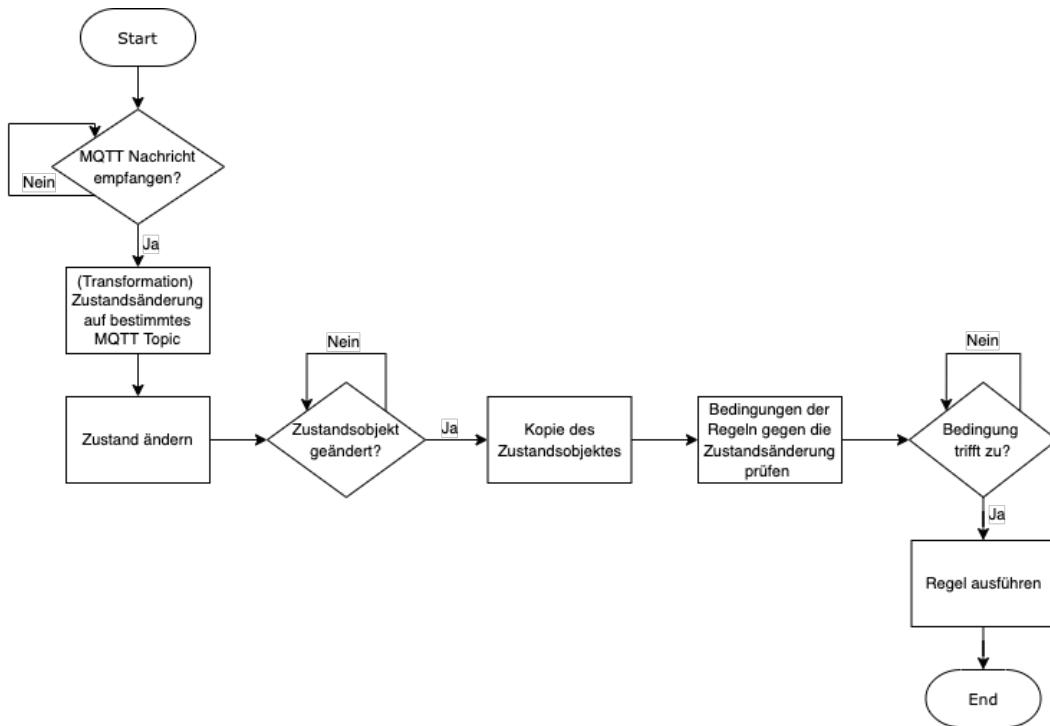
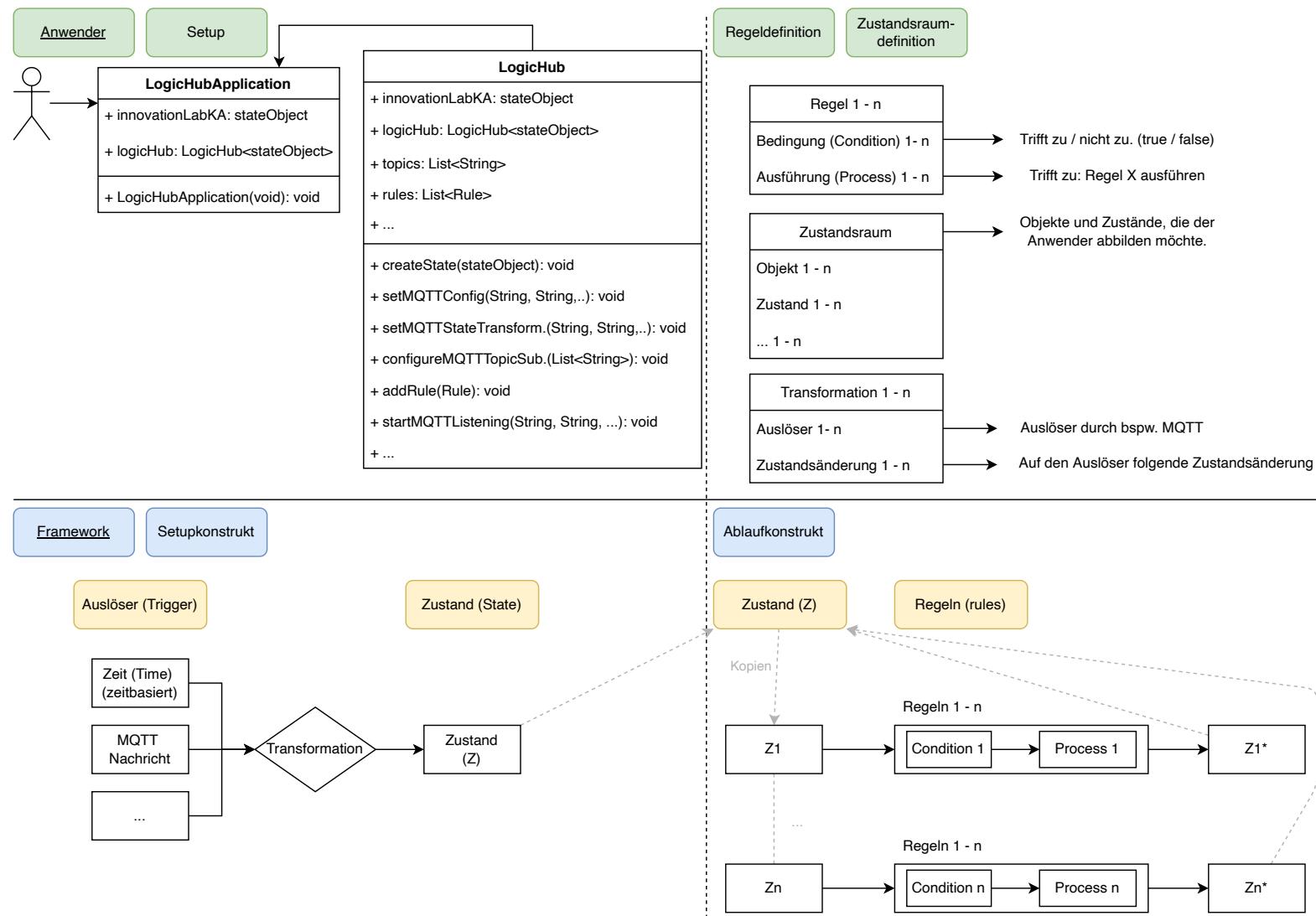


Abbildung 5.2: Grober Programmablauf des Frameworks

Aus dem Ablaufdiagramm gehen die parallele Ausführung von zwei voneinander unabhängigen Zustandsänderungen und die erneute Änderung des Zustandes durch eine Regel selbst nicht hervor. Sofern die Änderungen unabhängig sind, sollen Prozesse asynchron ablaufen, bzw. Regeln neue Zustandsänderungen hervorrufen können.

Zur Veranschaulichung hilft die nachfolgende Abbildung.



5.4 Architekturkonzept

Das Framework stellt die Kern-Funktionalität zur Steuerung von implementierten Regeln und Prozessen innerhalb eines Gebäudes, bspw. eines Bürouumes, dar. Mit dem Architekturkonzept wird der grundlegende Aufbau sowie die Funktionalität des Systems erläutert. Potentielle Erweiterungen und Adaptionen werden im Ausblick (9) aufgegriffen.

Das System soll in drei Schichten aufgeteilt werden. Die oberste Schicht stellt die Kommunikationsschicht dar, die mittlere die Logik- und Prozessschicht und die unterste die Persistenzschicht. Im Rahmen der Arbeit wird die dritte Schicht nicht detailliert beschrieben, da diese zum aktuellen Zeitpunkt weniger von Interesse ist, bzw. für keine weitere Verarbeitung genutzt wird. Die Ausprägung der Schicht wäre dennoch ohne weiteres möglich und wird grob skizziert. Der Abbildung (5.3) ist die Aufteilung der Schichtenarchitektur zu entnehmen. In der Darstellung unterscheidet sich die Persistenzschicht farblich von den anderen, da diese im aktuellen Konzept keine konkrete Implementierung erfährt. Das Schaubild zeigt bereits auf dieser Ebene die Trennung der Zuständigkeiten, engl. *separation of concerns*. Durch die gezielte Abstraktion von Komponenten und Informationen kann die Komplexität eines Systems vereinfacht dargestellt werden. Die Kommunikationsschicht ermöglicht die Anbindung von verschiedenen Kommunikationsprotokollen, die dadurch adaptiert werden können. Im Rahmen des Konzepts wird ausschließlich vom MQTT-Protokoll Gebrauch gemacht. Die Logikschicht nimmt alle eingehenden Events der Kommunikationsschicht, die jeweils eine Zustandsänderung hervorrufen können, entgegen und durchläuft den Prozess des Frameworks, um auf die Zustandsänderung die passende Regel auszuführen. Die Persistenzschicht ist für die Speicherung erzeugter Daten verantwortlich, bspw. können bei Zustandsänderungen die Transaktionen als Historie persistiert werden. Ebenso können Informationen gespeichert werden, die anderweitig zur Verfügung stehen würden. Das Vorantreiben der Speicherung von Informationen kann durch den agilen Entwicklungsprozess außerhalb des Thesis-Rahmens erfolgen.

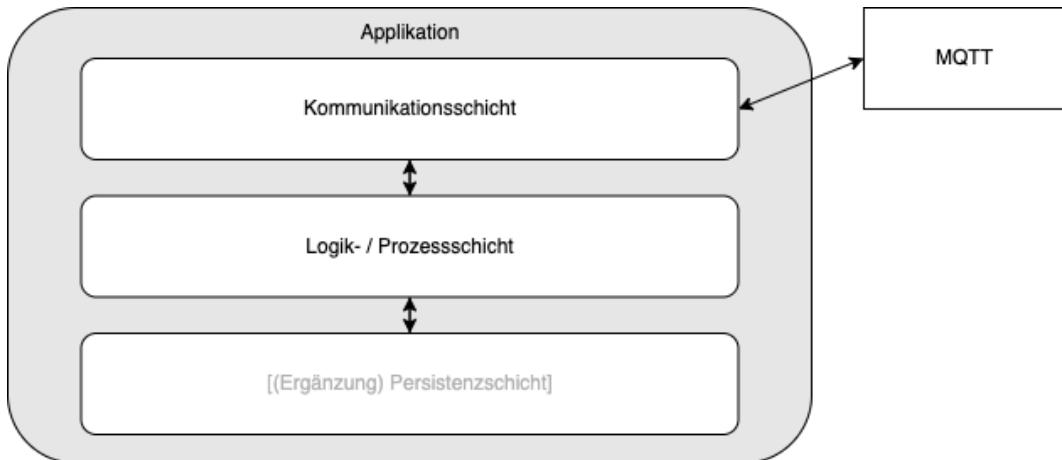


Abbildung 5.3: Schichtenarchitektur

Für die Konzeption von wartbarer und erweiterbarer Software wird von den *SOLID*-Prinzipien

sowie von nützlichen und für das Framework verwendbaren Programmiermustern Gebrauch gemacht. Diese werden an geeigneter Stelle und auf den Nutzen bezogen erläutert.

Das *SOLID* wurde von Robert C. Martin¹ geprägt und soll zum Design guter objektorientierter Software beitragen. Dadurch soll ein Softwaresystem einer höheren Wartbarkeit unterliegen. Es setzt sich aus fünf Prinzipien zusammen, die wie folgt definiert sind:

- S - Single-Responsibility-Prinzip: „Das Single-Responsibility-Prinzip (SRP; Eine-Verantwortlichkeit-Prinzip; ...) fordert, dass eine Klasse oder ein Modul einen und nur einen *Grund zur Änderung* haben sollte.“ [MARTIN 2009]
 - O - Open-Closed-Prinzip: „Module sollten sowohl offen (für Erweiterungen) als auch verschlossen (für Modifikation) sein.“ [MEYER 1988]
 - L - Liskovsches Substitutionsprinzip: „... Sei $q(x)$ eine beweisbare Eigenschaft von Objekten x des Typs T . Dann soll $q(y)$ für Objekte des Typs S wahr sein, wobei S ein Untertyp von T ist.“ [LISKOV und WING 2001] - Das (Ersetzungs-)Prinzip sagt somit aus, dass eine Subklasse stets die Eigenschaften der Superklasse erfüllt und immer als Objekt der Superklasse anwendbar sein muss. Eine Subklasse darf dadurch erweitert werden, aber keine grundlegende Änderung erfahren.
 - I - Interface-Segregation-Prinzip: „Clients sollen nicht dazu gezwungen werden, von Interfaces abzuhängen, die sie nicht verwenden.“ [MARTIN 1996]
 - D - Dependency-Inversion-Prinzip: „A. Module hoher Ebenen sollten nicht von Modulen niedriger Ebenen abhängen. Beide sollten von Abstraktionen abhängen. B. Abstraktionen sollten nicht von Details abhängen. Details sollten von Abstraktionen abhängen.“ [MARTIN 2003]

Besonders das *Single-Responsibility* und das *Open-Closed*-Prinzip wurden bei der Konzeption berücksichtigt. Da es sich bei diesem System nicht um eine klassische Webanwendung handelt und keine direkte Kommunikation unter anderem über REST APIs stattfindet, wird ein allgemeines Programmiermuster, wie bspw. das Model View Controller (MVC)² oder das Model View View-Model (MVVM)³, nicht in Betracht gezogen. Verwendet werden allgemeine Architekturmuster und -prinzipien, die dazu beitragen, die Software wartbar zu gestalten, die Komplexität einzelner Funktionen zu reduzieren und die Lesbarkeit und Überschaubarkeit zu erhöhen.

¹Software Engineer, Lehrer und Autor Robert C. Martin. https://en.wikipedia.org/wiki/Robert_C._Martin
Besucht am 10.07.2022

² Architekturmuster für Webapplikationen. https://de.wikipedia.org/wiki/Model_View_Controller Besucht am 10.07.2022

³Software-Architekturmuster. <https://en.wikipedia.org/wiki/Model\T1\textendashview\T1\textendashviewmodel> Besucht am 10.07.2022

5.4.1 Aufbau des Frameworks

Nach Veranschaulichung der Schichten-Modellierung werden übergreifend der Komponentenaufbau und die dafür einzusetzenden Architekturmuster aufgegriffen.

Die Konzeptkomponenten (5.3.1) finden sich im umfassenden Architekturnschaubild (siehe Abbildung 5.4) wieder. Die Regelstruktur soll die Funktion einer Bedingungsprüfung und der darauffolgenden Aktion vorgeben, damit bestimmte Anhaltspunkte geschaffen werden, die das Framework überprüfen und anwenden kann. Um zu gewährleisten, dass dem Anwender diese Methoden bei Implementierung einer spezifischen Regel zur Verfügung stehen, wird ihm mittels dem *Template Method* Pattern [GAMMA u. a. 1995a] (Schablonenmethoden-Entwurfsmusters) ein Leitfaden vorgegeben. Es gehört zu den Verhaltensmustern, engl. behavioral patterns. Ziel des Musters ist die Delegation der konkreten Ausformung einzelner Methoden und Funktionen zu deren Unterklassen. Innerhalb der abstrakten Klasse wird das Skelett des Objektes definiert. Die einzelnen Schritte können so in den darunter liegenden Klassen überschrieben oder ergänzt werden, ohne dass die grundlegende Struktur der übergreifenden Klasse modifiziert werden muss. Die Unterklassen, die konkret die Definition einer Regel ist, sollen unter Verwendung des *Singleton* Patterns [GAMMA u. a. 1995b] realisiert werden. Das zu den Erzeugungsmustern zugehörige Pattern dient zur Sicherstellung, dass eine Klasse, in dem Fall die einzelne Regel, zur Laufzeit ausschließlich eine einzige Instanz erzeugt und diese global erreichbar ist. Dadurch wird von jeder Regel immer nur ein Objekt erzeugt.

Ähnlich zum Aufbau des Regelwerks sollen die Komponenten, die reale Gegenstände abbilden, ebenso vom Schablonenmethoden-Entwurfsmuster Gebrauch machen. Unterschied zu der Regel-Schablone ist lediglich die Funktion der Sperrung einer Komponenten für weitere Aufgaben. Diese soll bereits in der abstrakten Klasse definiert sein. Konkrete Eigenschaften von Komponenten und konkrete Zustandsattribute können vom Anwender selbst implementiert werden.

Die durch den Anwender implementierte Transformation, die unter anderem basierend auf einem eingehenden MQTT-Topic das Zustandsobjekt verändert, soll das Template Method Pattern verwenden. Dadurch ist die Struktur der Transformation vorgegeben. Soll ein Transformationsobjekt erzeugt werden, so müssen diesem die notwendigen Parameter übergeben werden.

Damit zukünftig weitere Kommunikationsschnittstellen hinzugefügt werden können, soll für die Transformation zusätzlich das *Adapter* Pattern verwendet werden. Dadurch können Events aus unterschiedlichen Services und über mehrere Kommunikationswege entgegengenommen werden, ohne dass die eigentliche Schnittstelle modifiziert werden muss. Das *Adapter* Pattern gehört zu den Strukturmustern und ist wie folgt definiert:

„Das *Adapter* Pattern fungiert als Verbinde zwischen zwei inkompatiblen Schnittstellen, die ansonsten nicht direkt verbunden werden können. Ein Adapter umschließt eine vorhandene Klasse mit einer neuen Schnittstelle, sodass sie mit der Schnittstelle des Clients kompatibel wird.“⁴ [GAMMA u. a. 1995c]

⁴Beschreibung des Adapter Entwurfsmusters. <https://www.baeldung.com/java-adapter-pattern> Besucht am 10.08.2022

Das Zustandsobjekt ist aus Anwendersicht als ein einfaches Plain Old Java Object (POJO), ein ganz normales Java Objekt, zu implementieren. Das Framework soll sich um die korrekte Nutzung des Objektes kümmern.

Der übergreifende Aufbau unter Verwendung von Entwurfsmustern ist folgender Abbildung zu entnehmen:

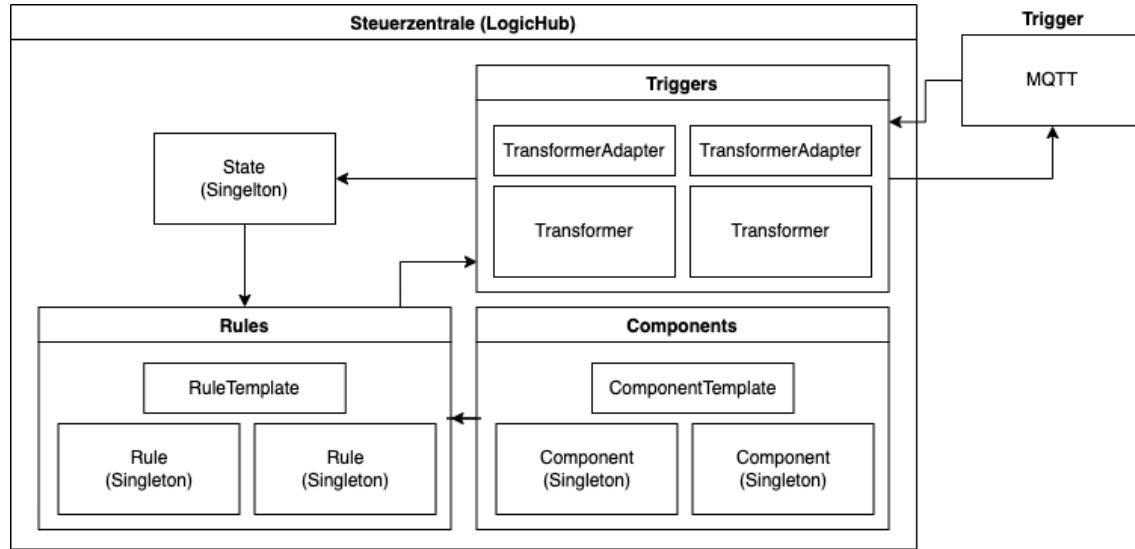


Abbildung 5.4: Architektur mit verwendeten Entwurfsmustern

Der soeben grob skizzierte Aufbau wird in den folgenden Abschnitten konkretisiert.

5.4.2 Kommunikationsschicht

Damit das Framework einer hohen Wartbarkeit und Erweiterbarkeit unterliegt, soll bereits in der Kommunikationsschicht mithilfe des genannten *Adapter* Musters eine Abstraktionsebene geschaffen werden. Diese erleichtert das Adaptieren hinzukommender Kommunikationsschnittstellen, um in Zukunft weitere IoT-Protokolle mit dem Framework zu nutzen. Die geplante Struktur kann dem Klassendiagramm in Abbildung (5.4) entnommen werden. Der Client, in dem speziellen Fall der *MQTT-Topic-Subscriber*, wird mit einem *Transformationsadapter* versehen. Hierbei wird mit den eingehenden Informationen, die über die Funktion des Subscribers abgegriffen und an das Frameworks übergeben wurden, gearbeitet. Wird eine weitere Kommunikationsschnittstelle hinzugefügt, so muss lediglich ein dafür vorgesehener *Adapter* implementiert werden, der alle für das Framework notwendigen Informationen delegiert. Dadurch bleibt die interne Arbeitsweise des Frameworks unverändert und fördert so die Erweiterbarkeit des Systems.

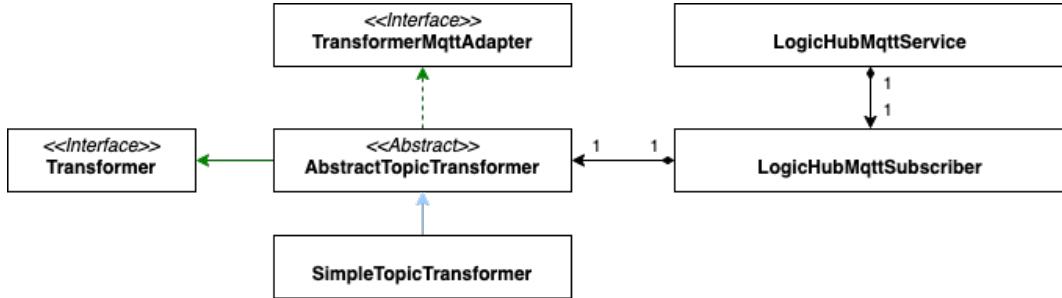


Abbildung 5.5: Klassendiagramm der Kommunikationsschicht

Der interne Ablauf einer Nachrichtenkonsumierung kann dem Sequenzdiagramm in Abbildung (5.6) entnommen werden. Mit der Transformation soll das eingehende MQTT-Topic mit den vom Anwender definierten Topics überprüft werden. Gibt es dabei eine Übereinstimmung, soll anhand des Topics die mitgelieferte Nachricht konsumiert und dem jeweiligen Wert im Zustandsraum zugeordnet und dementsprechend überschrieben werden. Mit der Zustandsänderung wird die Prozessschicht des Frameworks aufgerufen, die daraufhin alle dem Framework bekannten Regeln überprüft und die Zutreffenden startet. Der Durchlauf wird im Abschnitt (5.4.3) dargestellt.

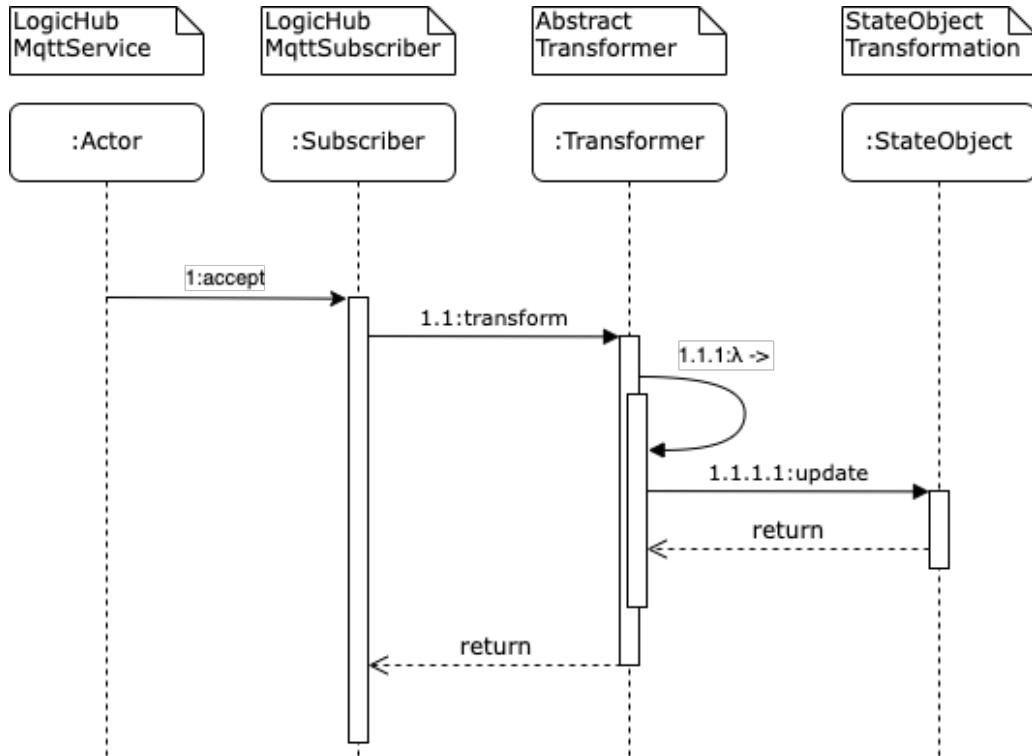


Abbildung 5.6: Sequenzdiagramm zur Kommunikationsschicht

5.4.3 Logik- und Prozessschicht

Die Logik- und Prozessschicht wird aktiv, wenn durch eine bestimmte Kommunikationsschnittstelle, im Rahmen der Arbeit ausschließlich durch MQTT realisiert, ein eingehendes Event, bzw. eine eingehende Nachricht konsumiert und daraufhin das Zustandsobjekt geändert wurde. Die in erster Linie dafür benötigten Objekte sind dem Klassendiagramm (5.7) zu entnehmen. Die an zentraler Stelle stehende Klasse des *LogicHubState* soll die Koordination der Zustandsänderung sowie das nachträgliche Prüfen der Regelbedingungen und das Ausführen des Regelprozesses übernehmen. Die *LogicHubState* Klasse muss eine *setState*-Methode enthalten, die die Schritte der Zustandsänderung und die des Kopiervorgangs des Zustandsobjektes beinhaltet. In der vorgesehenen Funktion wird anschließend mit der Kopie des Zustandsraumes und der hierin immanierten Änderung gearbeitet, indem daraufhin das Regelwerk durchlaufen wird und die Bedingungen der darin enthaltenen Regel überprüft werden. Durch die Verwendung der Kopie zum Durchlaufen der zutreffenden Regelprozesse entsteht keine Inkonsistenz des eigentlichen Zustandsraums. Somit wird verhindert, dass eine Zustandsänderung durch eine Regel das repräsentative Zustandsobjekt direkt manipuliert. Dadurch können Regelprozesse asynchron abgearbeitet werden und so mehrere Zustandsänderungen erfolgen. Zusätzlich wird der Kopiervorgang gesperrt, sodass keine Inkonsistenz des Zustandes durch gleichzeitig hereinkommende Änderungen entsteht.

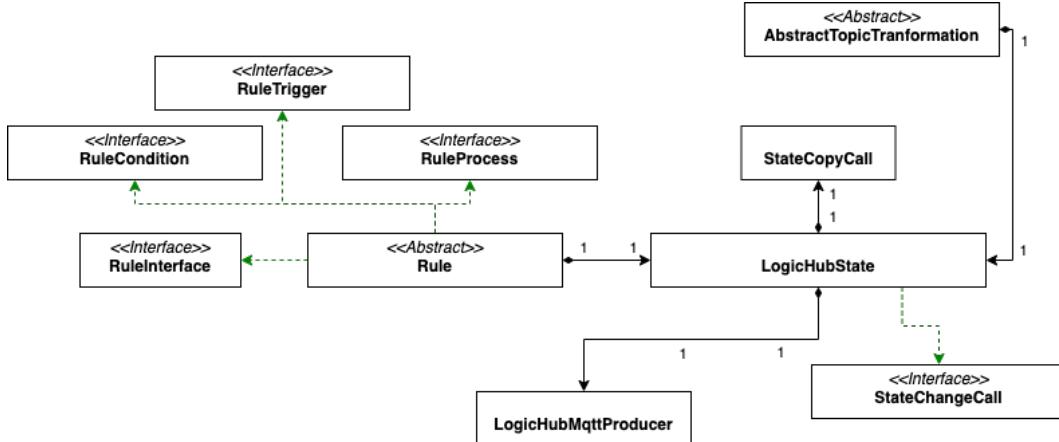


Abbildung 5.7: Klassendiagramm zur Logik- und Prozessschicht

Der initiale Ablauf von der Zustandsänderung bis hin zur Prüfung der Regelbedingung über die Erstellung einer Kopie sowie das anschließende Durchführen kann dem folgenden Sequenzdiagramm entnommen werden.

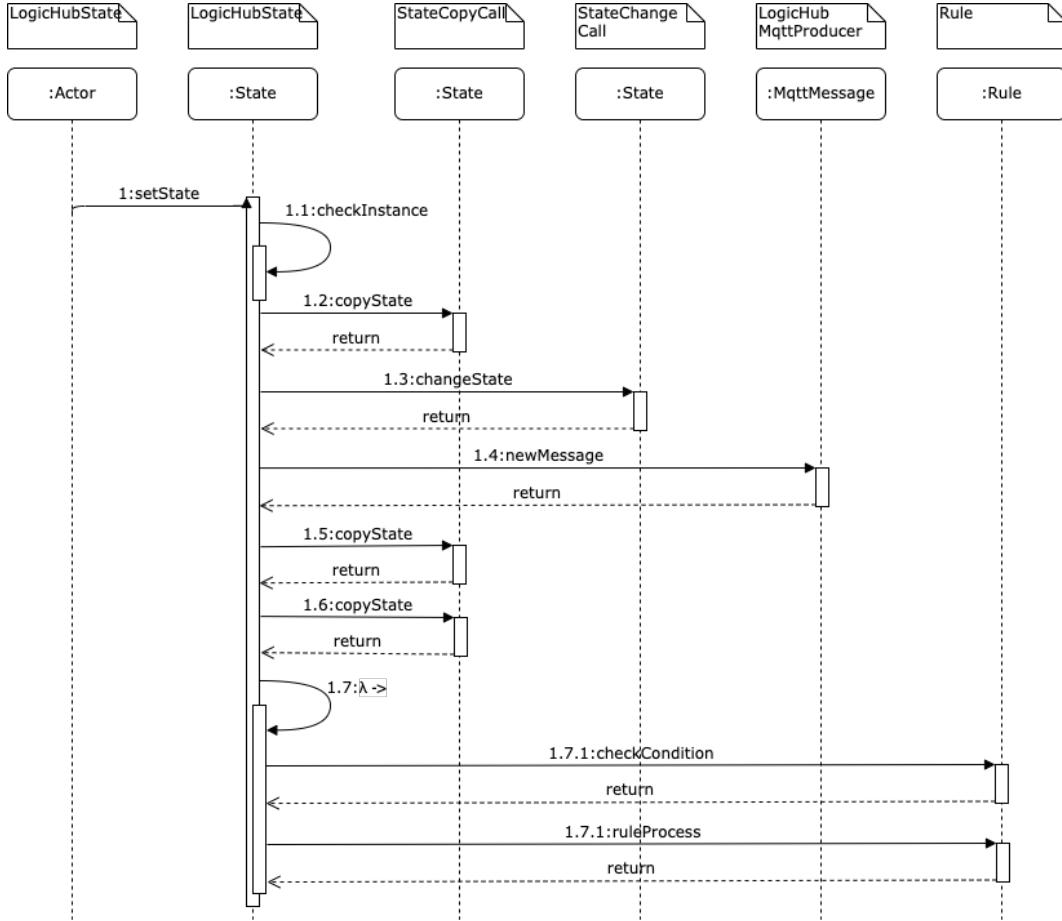


Abbildung 5.8: Sequenzdiagramm zur Logikschicht

Das Diagramm (5.8) beinhaltet ebenso die inverse Transformation (6.2.5). Mithilfe dieser sollen erneute Zustandsänderungen, die der Anwender durch eine Regel definiert, stattfinden, bspw. zur Veröffentlichung von Steuerbefehlen über eine MQTT-Nachricht. Auf die konkrete Umsetzung der soeben architektonisch beschriebenen Komponenten wird im folgenden Kapitel (siehe Umsetzung 6) eingegangen.

Die Ausprägung der Persistenzschicht wurde im Rahmen dieser Arbeit nicht weiter detailliert aufgezeigt, da zum aktuellen Zeitpunkt kein Bedarf an der Speicherung von Historien oder Zustandsänderungen besteht. Dennoch wird an dieser Stelle erwähnt, dass die Einbindung einer Datenbank sowie die Modellierung der Datenobjekte und deren Speicherung ohne weiteres möglich ist.

Für die Generierung mehrerer konkurrierender Sichten wurden die Diagramme, darunter Klassen- und Sequenzdiagramme (siehe 5.5, 5.6, 5.7 & 5.8) unter Verwendung des *4+1 Sichtenmodells* von Philippe Kruchten [KRUCHTEN 1995], erstellt. Die Beschreibung der Szenarien fand bereits in der Anforderungsanalyse (siehe 4.3.1 & 4.3.2) statt. Nach Beendigung der Konzeption wird im folgenden Kapitel die Umsetzung des Konzeptes und die Implementierung des Prototypen vertieft.

Kapitel 6

Umsetzung

Im Rahmen einer prototypischen Implementierung wurde das im Konzept (siehe Kapitel 5) geplante System umgesetzt. Gegenstand dieses Kapitels wird es sein, Aspekte der Umsetzung widerzuspiegeln und konkret einen Anwendungsfall darzulegen. Zusätzlich wird die Auswahl des dafür genutzten Frameworks aufgegriffen.

6.1 Auswahl des Frameworks

Im Bereich der Java-Entwicklung gibt es mittlerweile viele Möglichkeiten, Applikationen, Anwendungen und Frameworks zu entwickeln, die unterschiedliche Präferenzen bieten. Dadurch gibt es bezogen auf den Einsatzbereich im Vergleich zu ähnlichen Systemen Vor- und Nachteile. Eine kleine Auswahl an Frameworks wurden getestet und auf deren Brauchbarkeit analysiert und evaluiert. Für diese Evaluation wurden Kriterien ausgearbeitet, die die Auswahl des Frameworks einschränken, um nach Möglichkeit das Passendste herauszufiltern. Die Kriterien sind nach ihrer Relevanz aufgelistet:

1. Freiheiten bei der Nutzung (Anwendung und Gewährleistung von Entwurfsmustern)
2. Nutzung und Bereitstellung von Bibliotheken (Libraries)
3. Bereitstellung einer Plattform (Full Stack)
4. Aktive Community und stetige Weiterentwicklung des Systems
5. Nutzung des Frameworks in bereits bestehenden Projekten im Bereich Smart Home
6. Open Source-Software für Flexibilität und weitestgehende Unabhängigkeit
7. Ausschluss von Web-Frameworks

Nach Anwendung der oben aufgeführten Kriterien bleiben noch die Frameworks OSGi und Spring übrig. Die Frameworks, wie bspw. Grails, Quarkus, Blade, und Play, konnten das letzte Kriterium nicht erfüllen, bzw. basieren auf dem Spring-Framework.

6.1.1 OSGi

Das OSGi Framework der OSGi¹ Alliance, welches in der openHAB Software eingesetzt wird, ist eine dynamische Softwareplattform, mit der die Modularisierung und Verwaltung von Applikationen und den dazugehörigen Diensten mittels Komponentenmodell realisiert werden kann [FUNKE 2009]. Bekannte Produkte, die auf der OSGi Plattform laufen, sind neben openHAB unter anderem die Entwicklungsumgebung Eclipse der Eclipse Foundation, Produkte und Softwarelösungen von IBM, Oracle, Adobe und weitere.

Nennenswerte Eigenschaften und Vorteile der Software sind die Modularisierung und Versionierung, das zur Laufzeit organisierte Abhängigkeitsmanagement, das Fernmanagement des laufenden Systems über sogenannte Management Agents und die Nutzung des serviceorientierten Programmiermodells, Service-Oriented Architecture (SOA)². Die Funktionsweise und die technisch fundierte Erläuterung kann dem Buch [WUETHERICH u. a. 2008] sowie der Dokumentation [WUETHERICH u. a. 2009] eines ausgearbeiteten Workshops entnommen werden. Nachteile der Plattform sind zum einen der geringere Einsatz und zum anderen die kleine Community. Ebenso findet eine Weiterentwicklung der Plattform nur mäßig statt, was eine aufwändige Erweiterung, bzw. schleppende Entwicklung des darauf aufbauenden Systems zur Folge hat.

6.1.2 Spring

Das Ziel von Spring³ ist die Vereinfachung und Förderung von Programmierpraktiken in der Java- und Java EE Entwicklung. Mit einem breiten Spektrum an Funktionalitäten bietet das Framework eine ganzheitliche Lösung zur Entwicklung von Applikationen, Anwendungen und Frameworks. Im Vordergrund dabei steht immer die Entkopplung von Applikationskomponenten. Das Spring Framework wurde im März 2004 offiziell freigegeben und wird seitdem stetig weiterentwickelt. Initiator des Frameworks ist der Autor und Softwareentwickler Rod Johnson, der in seinem Buch folgende Prinzipien für das System verwendet [JOHNSON 2004]:

Dependency Injection ist ein sehr bekanntes Entwurfsmuster. Dabei werden die Abhängigkeiten eines Objektes zur Laufzeit reglementiert. Unter Verwendung des Frameworks werden diesen die benötigten Objekte und Ressourcen zugewiesen. Dadurch müssen sie nicht selbst gesucht werden.

Ein weiterer Punkt ist die aspektorientierte Programmierung, durch die der Entwickler technische Aspekte voneinander isolieren und den eigentlichen Programmcode von Transaktionen oder anderen Faktoren freihalten kann.

Das dritte Prinzip ist das Bereitstellen von Vorlagen zur Vereinfachung der Nutzung von Schnittstellen, sog. APIs. Dadurch wird ein POJO-basiertes Modell möglich.

Zusammengefasst sind beide Frameworks dazu geeignet, das Konzept der Steuerzentrale unterstützen.

¹ Ursprung der OSGi Plattform. <https://www.osgi.org/about/> Besucht am 19.06.2022

² Erläuterung des SOA Modells. <https://www.ibm.com/cloud/learn/soa> Besucht am 20.06.2022

³ Open-Source-Framework Spring. <https://spring.io> Besucht am 02.07.2022

zend zu realisieren. Ein direkter Vergleich zwischen den Frameworks kann nicht gezogen werden, da die Funktionalitäten, bzw. die aus der Nutzung profitablen Eigenschaften nicht vergleichbar sind. Während sich Spring auf die Unterstützung zur Erstellung von Applikationen konzentriert, legt OSGi den Mehrwert auf die Modularisierung durch Kontainerisierung. Durchaus ist eine Kombination aus beiden Frameworks möglich, um Vorteile beider zu vereinen. Grundlegend bietet Spring jedoch weitaus mehr Möglichkeiten und Unterstützungen zur Entwicklung von Applikationen und Frameworks, weshalb es im Rahmen dieser Arbeit eingesetzt wird. Mit Spring Boot, welches auf Spring aufbaut, können auch Model View Controller Web-Architekturen und REST APIs implementiert werden. So kann ein vollumfängliches System erstellt werden. Unter zusätzlicher Verwendung von OSGi wäre eine weitere Modularisierung und Kontainerisierung möglich. Dies ist auch durch Spring mit Hilfe von Microservices in einer anderen Form realisierbar.

Spring stellt ein modernes und etabliertes Framework dar, dessen Community erheblich größer ist als die des OSGi Frameworks. Hinzukommt die Schaffung eines alternativen Ansatzes zu openHAB, da dieses System bereits auf OSGi aufbaut. Dennoch unterscheidet sich die Steuerzentrale in den Funktionalitäten und Angeboten im Vergleich zu openHAB, denn diese bietet weitaus mehr Funktionalitäten, Erweiterungen und Ausprägungen. Lediglich kann die Definition von Regeln zum Vergleich herangezogen werden.

6.2 Implementierung

Hierbei wird konkret die Implementierung des Frameworks aufgegriffen und beschrieben, wie Konzeptentscheidungen im Code umgesetzt wurden. Auch werden wichtige Anhaltspunkte ergänzt, die in der Tiefe nicht aus dem Konzept hervorgehen, da sich dort lediglich auf die Idee und die allgemein gehaltene Vorstellung bezogen wurde.

6.2.1 Generische Programmierung

Ausschlaggebend für die Überlegung, die generische Programmierung, sog. *Generics*, zu verwenden, ist die Möglichkeit, mit einem Objekt zu arbeiten, ohne dass es vor der Systemlaufzeit bekannt ist. Der Entwickler kann dadurch über die Implementierung des Zustandsraums selbst entscheiden.

Ein Unterbegriff der *Generics* ist der *parametisierte Typ*, bei dem die Zuweisung von konkreten Werten zu einzelnen Parametern stattfindet. Mit der Nutzung von *Generics* ist ein syntaktisches Mittel gegeben, mit dem Klassen und Methoden mit Typen parametrisiert werden können. Diese Typ-Variablen sind zum Zeitpunkt der Implementierung unbekannt und können beliebig definiert werden. Erst zur Laufzeit des Systems wird die Typ-Variable durch ein konkretes Objekt ersetzt, wodurch dessen Struktur, Methoden und Funktionen bekannt werden. Bei der Verwendung von generischen Typen gibt es Varianzfälle, die unterschiedliche Auswirkungen aufzeigen. Der für das Konzept relevante Fall ist der des einfachen Typ-Parameters.

Ein konkretes Anwendungsbeispiel, wie es auch im Rahmen dieser Arbeit verwendet wird, sieht wie folgt aus:

Das Framework arbeitet mit dem generischen Typ E . Der Anwender soll nach der Implementierung der Klasse des Zustandsraumes das jeweilige Objekt dem Framework übergeben. So ist zur Laufzeit das konkrete Objekt bekannt. Zur Veranschaulichung dient folgendes Code-Beispiel:

```

1 public class LogicHub<E> {
2     private E state;
3     ...
4     public E getState() {
5         return this.state();
6     }
7 }

9 public class Application {
10     private final LogicHub<InnovationLab> logicHub;

12     public static void main(String[] args) {
13         logicHub = LogicHub.getInstance();
14         logicHub.getState(); // -> returns the state object
15                                         // and its values.
16     }
17 }
```

Code-Beispiel 6.1: Zustandsobjekt als Typ-Variable

Mit den Java Generics ist eine leistungsstarke Ergänzung der Java-Sprache gegeben, da es die Arbeit des Programmierers einfacher und weniger fehleranfällig macht. Zusätzlich wird durch die Generics die Typkorrektheit zur Kompilierzeit erzwungen und ermöglicht die Implementierung generischer Algorithmen, ohne für Anwendungen einen zusätzlichen Overhead zu verursachen.

Mit der Option des generischen Typs kann das Framework universell definiert und der Zustandsraum vom Anwender individuell implementiert werden.

Durch die Verwendung der Java Generics wird der konzeptionelle Schnitt zwischen Framework und Anwender-Interaktionen im Quellcode deutlicher.

6.2.2 Zustandsraum

Der vom Anwender zu beschreibende Zustandsraum stellt ein einfaches Java Objekt (*POJO*) dar, das in einer eigenen Klasse definiert wird. Die darin enthaltenen Felder und Attribute sollen reelle Gegenstände und deren Werte abbilden. Dies erfolgt durch Klassenvariablen, die sowohl einem primitiven Datentypen entsprechen als auch dem nicht primitiven *String*. Beispielsweise könnte eine LED-Leuchte als *boolean* mit den Werten *true* und *false* oder als *char* mit jeweils '*y*' und '*n*' oder mit dem nicht primitiven Typen *String* mit den Zeichenketten '*on*' und '*off*' definiert werden. Diese Entscheidung kann alleinig der Entwickler treffen.

Die im Zustandsraum erstellten Felder werden direkt an eine Transformation (siehe Abschnitt 6.2.4) gebunden. So kann sichergestellt werden, dass das Attribut im Zustandsobjekt immer den Wert erhält, der über die Transformation registriert, bzw. empfangen wurde. Dadurch kann auch

die Dopplung von Felder vermieden werden, was bedeutet, dass ein Feld nicht für mehrere reelle Gegenstände stehen kann. Für jedes Gerät ist mindestens ein eigenes Feld zu definieren. Durch die Verwendung von einfachen primitiven und nicht primitiven Datentypen innerhalb des Zustandsobjektes kann eine Rückkopplung umgangen werden.

Der Aufbau des *POJOs* entspricht einer einfachen Java-Klasse, die zusätzlich zu einem Konstruktor alle Felder, bzw. Attribute besitzt, die Zustände reeller Geräte abbilden sollen. Damit der Anwender in seinen definierten Regeln die Werte setzen und abfragen kann, gibt es zu jedem Attribut eine *Getter*- und *Setter*-Methode. Es ist wichtig anzumerken, dass das Zustandsobjekt mit dem *Singleton*-Pattern markiert und der Konstruktor dementsprechend initialisiert werden muss. Dadurch wird die Integrität des Objektes sowie die dadurch vermeidbare Redundanz gewährleistet. Anhand der Verwendung der *Lombok*-Bibliothek⁴ kann mittels der `@Data`-Annotation der (*boilerplate*) Quellcode reduziert werden. Zu Überprüfungszwecken wird mittels der Annotation das Objekt mit den einzelnen Feldern und Werten ausgegeben, wodurch das Objekt für den Entwickler lesbar wird. Das Konstrukt könnte in diesem Zusammenhang als Datenobjekt, bzw. als Datenmodell abgespeichert werden, um eine Transaktionshistorie der Zustandsänderungen zu erstellen.

Durch die Verwendung von Reflection (siehe Abschnitt 6.2.3) ist die Initialisierung des Zustandsraumes auf bestimmte Typendefinitionen beschränkt. Auf dieser Grundlage ist eine Verwendung von Objekten im Zustandsraum möglich, jedoch wird zum aktuellen Zeitpunkt davon abgeraten. Die Begründung dazu findet in den Modellierungsvorgaben (siehe 6.3.1) statt. Ebenso werden dort die Designvorgaben des Zustandsraums zum derzeitigen konzeptionellen Stand aufgegriffen.

Diese Entscheidung wurde getroffen, da in der Mehrheit der Anwendungsfälle die Zustände über primitive Typen abgebildet werden können. Ein Designnachteil ist dabei die Anlegung ggf. mehrere Attribute für ein Gerät im Zustandsraum, wodurch das Zustandsobjekt deutlich umfangreicher wird. Eine bessere Übersicht wäre gegeben, wenn die Felder in ein eigenes Objekt, bzw. eine eigene Klasse ausgelagert werden könnten. Der optische Nachteil ist dabei vertretbar. In weiteren Iterationen des agilen Vorgangs könnte eine weitere Alternative konzipiert werden, die Objekte im Zustandsraum zulässt. Im Rahmen dieser Arbeit war dies jedoch nicht vorgesehen.

Eine Aggregation von Objekten, bzw. deren Serialisierung ist nicht beabsichtigt, da die Transformation jeweils ein Attribut, bzw. Element bedient und somit eine eins-zu-eins-Beziehung zwischen dem Feld und dem MQTT-Topic besteht.

⁴Project Lombok. <https://projectlombok.org/features/Data> Besucht am 31.07.2022

6.2.3 Reflection

Mit der Reflection⁵ (Reflexion, Spiegelung) ist eine Funktion der Programmiersprache Java gegeben, die es ermöglicht, Laufzeitattribute von Klassen, Schnittstellen, Feldern und Methoden zu überprüfen und zu ändern. Diese Funktion erweist sich als besonders praktisch, wenn die Namen der Attribute zur Kompilierzeit nicht bekannt sind. Darüber hinaus können neue Objekte instanziert, Methoden aufgerufen und Feldwerte mithilfe von Reflexion abgerufen oder festgelegt werden.

Eingesetzt wird die Reflexion in der Transformation, die der Entwickler definiert, um aus eingehenden Events und Nachrichten Zustandsänderungen zu erzeugen. Konkret wird in der Transformation der Name des Feldes mitgegeben, der vom Entwickler bei der Definition des Zustandsobjektes vergeben wurde. Somit kann immer auf das Feld und dessen Wert zurückgegriffen werden und bei Bedarf durch eine eingehende Nachricht, bzw. durch eine Zustandsänderung innerhalb einer Regel überprüft und geändert werden. Die Struktur der Transformation wird im folgenden Abschnitt aufgezeigt.

6.2.4 Transformation

Die Transformation stellt die Schnittstelle zwischen den externen Auslösern, bspw. MQTT, und der Kommunikationsschicht dar. Die über das Kommunikationsprotokoll gelieferten Daten werden der *transform*-Methode übergeben, sodass die eigentliche Transformation stattfinden kann.

Im Rahmen der Thesis wurden zwei Lösungswege erarbeitet, die jeweils dem Entwickler ermöglichen, eine Transformation zu definieren. Diese werden beide erläutert und anschließend gegenübergestellt. Dem Anwender einen möglichst komfortablen Weg zur Implementierung einer Transformation mit geringem Aufwand zu bieten ist das dabei verfolgte Ziel.

Transformationsobjekt mittels Switch-Case-Anweisung

Die erste Option ist die Nutzung einer Switch-Case-Anweisung. Hierbei wird die *transform*-Methode bis zum Anwender durchgereicht. An dieser Stelle wird klar definiert, welches MQTT-Topic eine bestimmte Zustandsänderung auslöst. Innerhalb eines Falles (Case) wird über eine *Lambda*⁶-Funktion der jeweilige Wert im Zustandsobjekt geändert. Durch den direkten Aufruf der Methode zur Zustandsänderung wird mit dem neuen Wert die Regelüberprüfung und -ausführung ausgelöst. Das konkrete Vorgehen ist dem folgenden Sequenzdiagramm sowie dem Ausschnitt des Quellcodes (siehe 6.2) zu entnehmen.

⁵ Java Reflection. <https://www.oracle.com/technical-resources/articles/java/javareflection.html>
Besucht am 27.07.2022

⁶ Java Lambda-Funktion. <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
Besucht am 28.07.2022

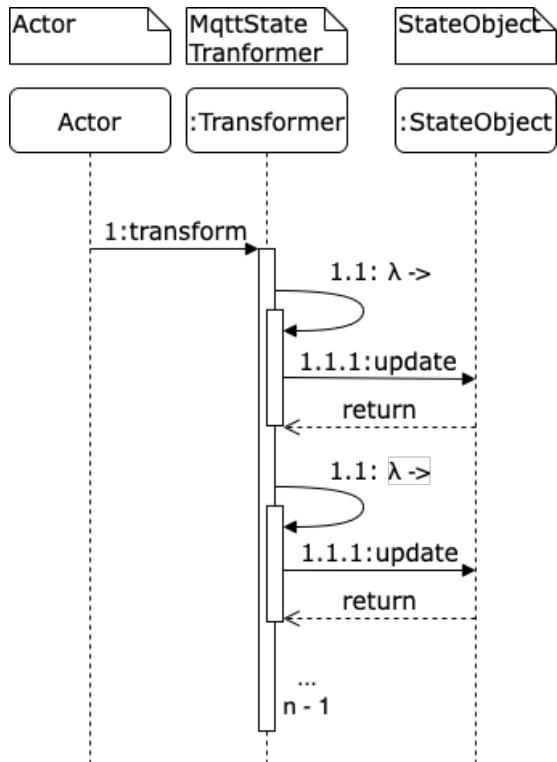


Abbildung 6.1: Transformation über Switch-Case-Anweisung

```

1 public class MqttTransformer extends AbstractTransformer<InnovationLabKA> {
2     private final LogicHubState<InnovationLabKA> logicHubState;
3
4     public void transform(String topic, String payload) {
5         switch (topic) {
6             case "InnovationLab/LogicHub/PersonOnDoor":
7                 logicHubState.setState((innovationLab) -> {
8                     innovationLab.setPersonOnDoor(payload);
9                     innovationLab.update();
10                });
11                break;
12            case "InnovationLab/LogicHub/SR/GoTo":
13                logicHubState.setState((innovationLab) -> {
14                    innovationLab.setSRGoTo(payload);
15                    innovationLab.update();
16                });
17                break;
18        }
19    }
20}

```

Code-Beispiel 6.2: Transformeation über eine Switch-Case-Anweisung

Dieser Ansatz gibt eine Übersicht über die Topics und deren dafür vorgesehenen Zustandsänderungen. Nachteile dabei sind jedoch die Zunahme der Anweisungen, d.h. je mehr Regeln und dafür vorgesehene Zustandsänderungen hinzugefügt werden, desto unübersichtlicher und länger wird die Switch-Case-Funktion, und die Kenntnis der MQTT-Topics an verschiedenen Stellen der Anwenderinteraktionen. Um innerhalb einer Regel eine weitere Zustandsänderung vornehmen zu können, muss der Anwender das Topic erneut aufgreifen, damit über den MQTT-Producer eine Nachricht veröffentlicht und eine Aktion ausgelöst wird.

Dieser unpraktische Lösungsweg galt es dem Anwender zu erleichtern, weshalb ein zweiter erarbeitet wurde, der im folgenden Abschnitt aufgezeigt wird.

Transformationsobjekt mittels eigenem Objekt

In Opposition zu der Verwendung einer Switch-Case-Anweisung wird bei diesem Ansatz eine Klasse implementiert, mit der je nach Bedarf weitere Objekte mit unterschiedlichen Werten erzeugt werden können. Dabei werden in dem Konstrukt der Klasse die jeweiligen Parameter definiert. Das zu erzeugende Objekt wird dem Framework zur Laufzeit übergeben und in einer Liste gespeichert, damit auf dieses zurückgegriffen werden kann. Die Parameter, die der Entwickler bestimmt, haben folgende Funktion:

- 1. Parameter: Hier wird das Topic übergeben. Der MQTT-Subscriber prüft, ob das konsumierte Topic einem in der Liste definierten entspricht.
- 2. Parameter: Hier wird der Key definiert. Das über das Topic mitgelieferte JSON-Konstrukt wird auf diesen Key überprüft. Dessen Wert ist für den Zustandsraum relevant. Hierüber wird der Zustand des Objektes transportiert.
- 3. Parameter: Hier wird optional eine Liste übergeben, die nur bestimmte Werte für den Key zulässt, eine sogenannte *White-List*.
- 4. Parameter: Hier wird der Name des Feldes im Zustandsraum wiedergegeben, sodass mittels Reflection der Wert im Zustandsraum adressiert werden kann.
- 5. Parameter: Hier wird optional der Name des Komponentenfeldes im Zustandsraum wieder-gegeben, sodass mittels Reflection der Komponentenwert im Zustandsraum adressiert werden kann.

Diese Werte bilden die Transformation, die ebenso für die Inverse (siehe Abschnitt 6.2.5) gilt. Der Ablauf kann dem folgenden Sequenzdiagramm entnommen werden:

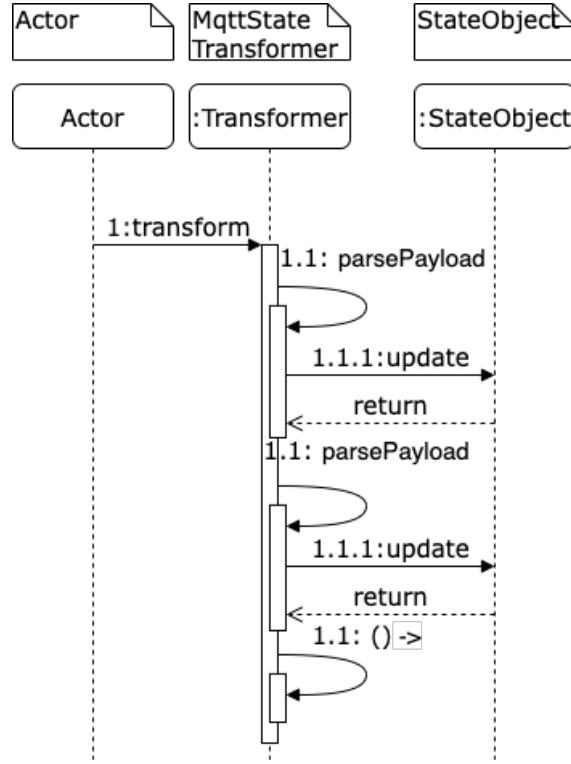


Abbildung 6.2: Transformation über eigenes Objekt

Um das Objekt zu erzeugen, muss der Entwickler lediglich das folgende Code-Beispiel verwenden und auf die entsprechenden Werte anpassen:

```

1 public class InnovationLogicHubApplication {
2     public InnovationLogicHubApplication() {
3         ...
4         logicHub.addTransformation(
5             new SimpleTopicTransformation(
6                 "InnovationLab/LogicHub/PersonOnDoor",
7                 "email",
8                 "personOnDoorEmail"
9             )
10        );
11    }
12 }

```

Code-Beispiel 6.3: Transformation über ein eigenes Objekt

Für die jeweiligen Konfigurationsmöglichkeiten der Transformation existieren drei Konstrukte. Neben der im Code-Beispiel (6.3) aufgeführten gibt es die Struktur mit dem zusätzlichen Parameter der *White-List*. Die dritte Möglichkeit spiegelt zusätzlich den Namen eines Feldes innerhalb einer Komponente wieder. So können Objekte theoretisch in den Zustandsraum aufgenommen werden,

wovon allerdings zum aktuellen Zeitpunkt abgeraten wird. Die Begründung zu dieser Modellierung wird in dem Abschnitt (6.3.1) aufgegriffen.

Der interne Ablauf des Frameworks unterscheidet sich bei beiden Transformationsmethoden in Nuancen, wie den beiden Sequenzdiagrammen (6.1 und 6.2) zu entnehmen ist. Jedoch wird bei dem zweiten Ansatz das Zustandsobjekt mittels Reflection manipuliert und anschließend als Änderung registriert und ausgeführt, während bei der ersten Methode die Änderung direkt über die Lambda-Funktion gesteuert und ausgelöst wird. Dies hat für den Anwender keinerlei Auswirkung auf die Funktionsweise des Frameworks. Innerhalb dieser Thesis war das Ziel die einfache Handhabung der formalisierten Interaktionen und die übersichtliche Gestaltung. Der zweite Ansatz liefert dem Entwickler diese Vorteile und wird deshalb angewendet.

Die Funktionsweise der Transformation ist demnach wie folgt umgesetzt:

Die vorab definierten Transformationen des Entwicklers werden in einer Liste gespeichert, damit das darin enthaltene Topic mit dem vom MQTT-Subscriber eingehenden verglichen werden kann. Dafür wird die Liste wiederholt überprüft sowie jede Transformation und deren zugewiesenes Topic. Sofern es eine Übereinstimmung gibt, wird der Key-Wert ermittelt und als Nutzlast (*Payload*) übergeben. An dieser Stelle wird der *ReentrantLock* aktiviert und die Feldmanipulation für den Wert, der mittels Nutzlast mitgegeben wurde, durchgeführt. Nachdem die Zustandsänderung erfolgt ist, wird der *Lock* deaktiviert und die Zustandsänderung an die Logikschicht weitergegeben. Somit ist die Transformation für ein eingehendes MQTT-Topic abgeschlossen. Der Ablauf kann dem Sequenzdiagramm (siehe Abbildung 5.6) entnommen werden.

Die Funktion der inversen Transformation wird im folgenden Abschnitt erläutert.

6.2.5 Inverse Transformation

Damit der Anwender des Frameworks bei der Regeldefinition einen geringen Aufwand hat und dennoch dadurch eine weitere Zustandsänderung hervorrufen kann, findet an dieser Stelle eine inverse Transformation statt. Dadurch wird anhand der Zustandsänderung, die der Entwickler durch eine Funktion in der jeweiligen Regel vorgibt, die zugehörige Aktion mittels MQTT-Nachricht veröffentlicht. In Folge dessen muss der Entwickler bei der Implementierung einer Regel keine Kenntnis über das MQTT-Topic haben, lediglich muss das richtige Attribut im Zustandsraum angesprochen und geändert werden. Ein konkretes Beispiel:

Wird ein Lichtschalter betätigt, wird daraufhin ein MQTT-Topic veröffentlicht und unmittelbar von der Steuerzentrale konsumiert. Dadurch wird der Zustandsraum über eine Transformation geändert. Eine Regel, deren Regelaktion die LED-Leuchte aktiviert, wird ausgeführt. Dementsprechend definiert der Entwickler eine Funktion, die den Wert des Lichtzustandes im Zustandsobjekt verändert. Mit der Veränderung wird überprüft, um welches Attribut es sich im Zustandsraum handelt und wie sich der Wert geändert hat. Wenn diese Bedingung zutrifft, wird im Rahmen der inversen Transformation ein *JSON*-Konstrukt generiert, welches mit dem in der Transformation definierten Topic als MQTT-Nutzlast an den MQTT-Broker publiziert wird. Dabei handelt es sich konkret um das Topic, auf das der MQTT-Client der LED-Leuchte reagiert. Daraufhin wird die LED-Leuchte angeschaltet.

Zusammengefasst wird mit der inversen Transformation vermieden, dass der Entwickler sich bei der Regelimplementierung neben der zu definierenden Zustandsänderung zusätzlich um das veröffentlichen der jeweiligen Nachricht und das Verpacken in eine übertragbare *JSON*-Struktur kümmern muss. Des weiteren ist die Kenntnis über das Topics, unter dem die Nachricht gesendet wird, ausschließlich in der Transformation von Nöten und muss nicht an mehreren Stellen konkret aufgeschrieben werden. Dafür kann zu jeder Zeit die Liste der Transformationen genutzt werden, um das erforderliche Topic zu erhalten.

Zum aktuellen Zeitpunkt wird bei der inversen Transformation ein einfaches *JSON*-Konstrukt erzeugt. Dafür wird der *Key*-Wert der Transformation und die Nachricht als Nutzlast verwendet. Zur Veröffentlichung der Nachricht unter einem MQTT-Topic wird derzeit ausschließlich der *Zigbee2MQTT*-Standard⁷ eingehalten. Dabei ergänzt sich die Topic-Zeichenkette, die über das Transformationsobjekt abgerufen werden kann, um den Anhang (*/set*). Hierfür wird vorab das *JSON*-Konstrukt generiert, mit den jeweiligen Werten belegt und als Nutzlast mittels dem Topic und dessen Anhang herausgegeben. Soll im Rahmen der Veröffentlichung einer Nachricht kein *JSON*-Konstrukt sondern ein einzelner Wert publiziert werden, so kann dies erfolgen, indem das Topic um den zugehörigen *Key*-Wert der Transformation (*/set/[Key-Wert]*) ergänzt wird. Ein Beispiel dazu: Eine LED-Leuchte wird mit einer Transformation mit den Werten (*'mqtt/topic/test'*), *'key-state'* und dem Attribut *'led-leuchte'* definiert. Soll in dem Zuge die Nutzlast nur den Wert *'on'* enthalten, so wird kein *JSON*-Konstrukt erzeugt, sondern lediglich die Nachricht über das modifizierte Topic (*'mqtt/topic/test/set/key-state'*) veröffentlicht.

Resümierend gibt es im Kontext des *Zigbee2MQTT* zwei Wege, um mit Geräten, die das ZigBee-Protokoll unterstützen, zu kommunizieren. Diese sind beide über das Framework, somit auch über die inverse Transformation abgedeckt.

6.2.6 Regeldefinition

Damit die Vorgaben für eine valide Regeldefinition eingehalten werden, wird dem Anwender mittels *Template Methode*-Pattern eine Schablone vorgeschrrieben, die alle notwendigen Methoden bestimmt, die der Entwickler implementieren sollte. Zusätzlich gibt es eine Annotation, die *RuleAnnotation*, die eine Regel als solche kenntlich macht und die Implementierung der Methoden forciert. Die beiden wichtigsten Funktionen innerhalb der Klasse, die durch die abstrakte Definition zum Softwareentwickler durchgereicht werden, sind zum einen die *checkCondition*-Funktion und zum anderen die *ruleProcess*-Funktion. Innerhalb der erstgenannten Methode gibt der Anwender einen Sachverhalt vor, der zutreffen muss, sodass anschließend diese Regel ausgeführt werden kann. Die Bedingung ist abhängig von dem Feld des Zustandsraumes und der Absicht, für die der Regelprozess gestartet werden soll. Sofern die Bedingung zutrifft, wird ein boolescher Wert (*true*) zurückgeliefert, der die Regel ausführen lässt. Die ebenso wichtige Funktion des Regelprozesses beinhaltet die Schritte der Regelausführung als klassische Methode, die keinen konkreten Wert zurückgibt, sondern lediglich

⁷Nutzung von *Zigbee2MQTT*. https://www.zigbee2mqtt.io/guide/usage/mqtt_topics_and_messages.html
Besucht am 30.07.2022

die darin enthaltenen Punkte abarbeitet. An dieser Stelle können wiederum weitere Zustandsänderungen implementiert werden. Dies ist abhängig von dem Inhalt der Regel und liegt ebenso in der Verantwortung des Entwicklers.

Neben den beiden Hauptfunktionen gibt es eine weitere Funktion, die den Auslöser der Regel definiert. Diese dient dazu, um Regeln ggf. nur dann anstoßen zu können, wenn ein bestimmter Auslöser dafür aktiviert wurde. Zusätzlich gibt es weitere Funktionen, die reine Informationswerte enthalten, wie den Regelnamen und die -beschreibung. Darüber hinaus gibt es für den Entwickler keinerlei Einschränkungen das Regelobjekt um weitere Funktionen zu ergänzen.

6.2.7 Parallelisierung

Bereits im Konzept wurde der Terminus der Parallelisierung und der Asynchronität aufgegriffen. Für das Auslösen einer Regel wird der gesamte Prozess der Bedingungsprüfung und der anschließenden Ausführung in einen *Thread* ausgelagert, der durch einen *ThreadPoolExecutor*⁸ im Java Code angestoßen wird. Es können durch mehrere *Threads*, die in sich asynchron arbeiten, mehrere Regeln gleichzeitig ausgeführt werden, sofern die Bedingungen dafür zutreffen.

Sowohl die parallele als auch die sequenzielle Ausführung können innerhalb der Regel Zustandsänderungen hervorrufen.

Eine sequenzielle Ausführung der Regeln ohne Verwendung von *Threads* wäre sehr ineffektiv und langsam, da eine Regel erst dann ausgeführt wird, wenn sie an der Reihe ist. Der Anwender weiß nicht, zu welchem Zeitpunkt die gewünschte Aktion eintritt, da er die Anzahl der dazwischenliegenden Sequenzen nicht kennt. Wenn bspw. ein Lichtschalter betätigt werden würde und davor noch zwei Regeln abgearbeitet werden müssten, würde dies Zeit in Anspruch nehmen, wodurch sich das Leuchten der Lampe verzögert.

In dieser Arbeit wird die parallele Ausführung bevorzugt. Jedoch ist es durch die Parallelisierung notwendig, die innerhalb der *Threads* eventuell entstehenden Zustandsänderungen anschließend zusammenzuführen, indem der Wert auf das eigentliche Objekt übertragen wird. Trotz der hinzukommenden Komplexität durch die Zusammenführung überwiegt der Vorteil, da die gewünschte Aktion unmittelbar ausgeführt wird, unabhängig von den derzeit parallel laufenden Prozessen. Damit die Zustandsänderung und die darauf zu erstellende Kopie entsprechend sauber durchgeführt wird, wird dieser gesamte Vorgang durch einen Lock-Mechanismus, der durch einen *ReentrantLock*⁹ realisiert ist, gesperrt. Dadurch ist sichergestellt, dass die Änderung sowie die Kopie für die weiteren Schritte tatsächlich zur Verfügung steht.

⁸ Spezifikation des Thread-Pools. <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html> Besucht am 27.07.2022

⁹ Spezifikation des Locks. <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/ReentrantLock.html> Besucht am 27.07.2022

6.2.8 Regelwerk -und management

Anknüpfend an die abgeschlossene Transformation wird der Teil ausgeführt, der basierend auf der Zustandsänderung alle bekannten Regeln überprüft und die zutreffenden abarbeitet. Wenn eine Zustandsänderung keiner konkreten Regel zugeordnet werden kann, fehlt diese entweder, wurde davor schon ausgeführt oder der Entwickler beabsichtigt keine Ausführung und sie wird ohne Auswirkung übertragen.

Damit jedoch voneinander unabhängige Regeln parallel ausgeführt werden können, wird der Prozess der Regelüberprüfung und -durchführung in einen *Thread* ausgelagert, wie bereits im Abschnitt der Parallelisierung (siehe 6.2.7) beschrieben. Der Ablauf kann dem Sequenzdiagramm (siehe Abbildung 5.8) ab dem Schritt 1.5:*copyState* entnommen werden.

Nach Implementierung des Frameworks sind alle Grundlagen des Systems zur Umsetzung der Use Cases gelegt und kann auf seine Funktionalität getestet werden, um Anhaltspunkte für die Evaluation (siehe Kapitel 7) zu schaffen. Der folgende Abschnitt behandelt die praktische Umsetzung zweier Anwendungsfälle.

6.2.9 Implementierung von Anwendungsfällen

Zur Überprüfung des Frameworks wurden im Rahmen der Arbeit drei Anwendungsfälle implementiert, um anhand deren die Funktionsweise darzustellen. Zum einen wurde die Steuerung einer LED-Lampe über einen Schalter und ein Steckdosen-Funkmodul umgesetzt, um einen Vergleich zwischen der Regeldefinition in Home Assistant, openHAB und der Steuerzentrale (InnovationLogicHub) ziehen zu können. Zum anderen wurden die beiden Anwendungsfälle, die im Rahmen der Anforderungsanalyse (siehe Kapitel 4) als Anwendungsfälle (4.3) definiert wurden, realisiert.

Lichtregelung

Der Lichtschalter-Anwendungsfall besteht darin, dass mittels einem physischen oder virtuellen Schalter eine LED-Leuchte eingeschaltet werden kann. Zur Gegenüberstellung der Umsetzungen wurden jeweils die gleichen Komponenten im selben Netzwerk verwendet. Dadurch sind die identischen Voraussetzungen geschaffen. Zu Beginn musste sichergestellt werden, dass die Komponenten mindestens das *ZigBee*-Kommunikationsprotokoll unterstützen, damit über *Zigbee2mqtt* die Nutzung von MQTT möglich ist.

Um eine Grundlage für die Analyse und Untersuchung der jeweiligen Plattformen zu legen, wurden im Rahmen dieser Arbeit zu Anfang bereits Instanzen von Home Assistant und openHAB gestartet. Diese wurden zur Umsetzung des einfachen Anwendungsfalls unter anschließender Beurteilung der Resultate genutzt. Die Integration der Elemente in die jeweilige Plattform wird an dieser Stelle nicht weiter erläutert, zur Veranschaulichung wird lediglich die Regeldefinition aufgegriffen. Im Fokus steht jedoch die Umsetzung über die Steuerzentrale.

Unter Anwendung des Frameworks werden im folgenden Abschnitt die Schritte dargestellt, die für die Realisierung des Anwendungsfalls notwendig sind:

Damit die Zustände der Komponenten abgebildet werden konnten, mussten diese im Zustandsraum

hinterlegt werden. Da der Schalter sowie das Funkmodul beide die Werte '*on*' und '*off*' über das Kommunikationsprotokoll versenden, wurde zur Abbildung der Zustandsattribute jeweils der *String*-Datentyp gewählt. Durch die Nutzung der *lombok*-Bibliothek sind die *Getter*- und *Setter*-Methoden gegeben.

Anhand der definierten Zustandsattribute konnten anschließend die Transformationen für beide Komponenten implementiert werden. Dafür notwendig waren jeweils die MQTT-Topics, der Name des Wertes innerhalb des *JSON*-Konstruktes, ggf. eine Liste der akzeptierten Zustandswerte und der Name des im Zustandsraum definierten Feldes. Die Transformationen waren daraufhin dem Framework zu übergeben, damit diese in die dafür vorgesehene Liste aufgenommen und zur Laufzeit als Objekte erzeugt werden konnten. Abschließend war die Regel zu definieren, darunter die zu prüfende Bedingung und die darauffolgende Aktion. Diese beinhaltet die Änderung des Zustandsattributes auf den Wert des Schalters, worauf über die inverse Transformation die MQTT-Nachricht publiziert wird, die an das Funkmodul gerichtet ist. Die beiden Methoden sind überschaubar und mit einem Quellcode weniger als fünf Zeilen zu implementieren. Dem Ausschnitt (siehe Anhang F.1) sind die Methoden zu entnehmen.

Die Umsetzung des Anwendungsfalls in Home Assistant stellte sich als umständlicher und aufwändiger dar. Hierbei ist im Kontext des Home Assistant eine übergreifende Automation, zwei Szenen (Aktionen) für das Schalten des Lichts und ein Auslöser für jede Aktion des Schalters zu definieren. Die Konfigurationsdatei erstreckt sich über 50 Zeilen, in denen die eigentliche Funktion eingefügt wird. Die Benutzeroberfläche gibt dem Anwender die Möglichkeit, anhand der Informationen die Konfigurationsdatei zu erzeugen. Dadurch ist der Entwickler nicht an die *YAML*-Struktur gebunden und kann ausschließlich die Werte, um die es sich handelt, eintragen. Das Erstellen der Automatisierung über die Nutzeroberfläche erschließt sich dem Anwender von der Logik her einfacher, ist dennoch aufwändiger durch viele Interaktionen. Ergänzend dazu ist dem Anhang ein Ausschnitt der Konfigurationsdatei beigelegt (siehe Anhang F.2).

Ähnlicher Konfigurationsaufwand war bei der Umsetzung in openHAB über die Benutzeroberfläche festzustellen. Das skriptbasierte Definieren von Regeln erfordert eine längere Einarbeitungszeit, da die Semantik des *ECMAScripts* oder die Funktionsweise von bspw. *Blockly* erst erlernt und verstanden werden muss. Die Regeldefinition mittels der *Rule DSL* hingegen kann mit der Steuerzentrale verglichen werden. Diese ist in ähnlich wenigen Schritten realisiert. Der Quellcode-Ausschnitt zeigt den Umfang der Regel über die openHAB DSL (siehe Anhang F.3).

Die folgenden Anwendungsfälle wurden ausschließlich über die Steuerzentrale umgesetzt.

Check-in mit einem Service-Roboter

Die Handlungsschritte, um einen Anwendungsfall mit dem Framework zu realisieren, gleichen sich grundlegend. Damit alle für den Fall notwendigen Zustände abgedeckt sind, wurden diese in dem Zustandsraum hinterlegt. Zu den notwendigen Attributen zählen unter anderem die E-Mail-Adresse, bzw. der Name der authentifizierten Person, die Verfügbarkeit des Service-Roboters, sowie dessen ausführbare Prozesse. Anschließend wurden die Transformationen zur Erzeugung der Schnittstelle (siehe Kapitel 6.2.4) definiert.

Sobald eine Person über die Kamera authentifiziert wurde, wird ein MQTT-Topic gesendet, welches als Nutzlast ein JSON-Konstrukt mit einem Zeitstempel, den Vor- und Nachnamen, sowie die E-Mail-Adresse der Person übergibt. Anhand dessen wird im Zustandsraum der Wert der E-Mail geändert, woraufhin die Regeln auf ihre Bedingung überprüft werden. Dabei wird die Regel zum Grüßen und Einchecken des Gastes gestartet. Hierfür wird der Service-Roboter an die Tür geschickt, an der er den Gast empfängt und begrüßt. Während der Roboter an die Tür fährt, checkt er die Buchungen und überprüft, ob die Person einen Platz gebucht hat. Nachdem der Gast empfangen und eine Buchung gefunden wurde, checkt der Roboter die Person ein und gibt daraufhin Rückmeldung, dass die Person erfolgreich im Büroplatzbuchungssystem eingecheckt wurde. Ist der Gast bereits angemeldet oder hat vorab keine Buchung getätigt, wird er aktuell darum gebeten, nachträglich einen Platz zu buchen und darauf einzuchecken.

Die Regelabfolge kann zusätzlich dem Sequenzdiagramm (siehe Abbildung 6.3) entnommen werden.

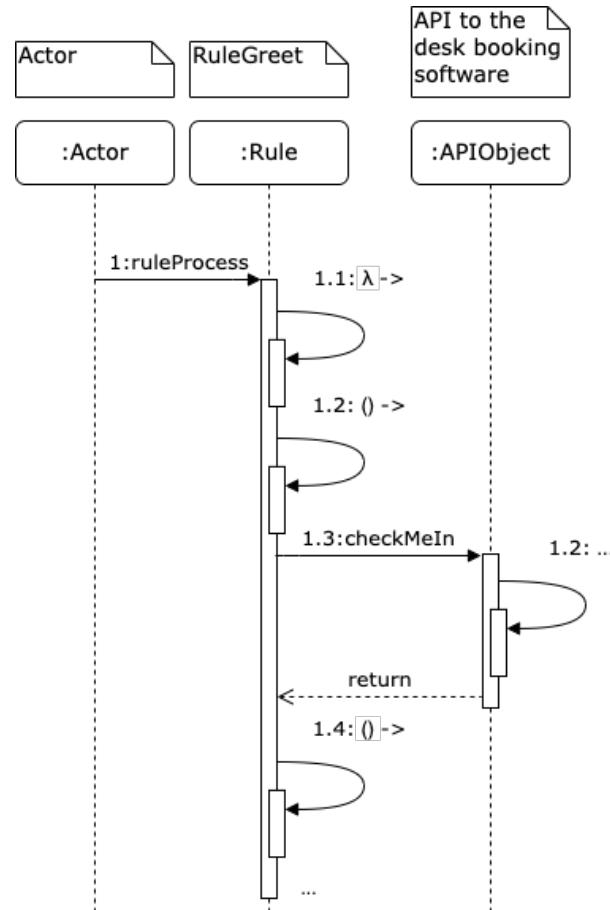


Abbildung 6.3: Check-in Regelablauf (RuleProcess)

Der Check-in wurde im Rahmen dieser Arbeit als eine Handlungskette implementiert, da der Roboter zu aktuellem Zeitpunkt noch keine interaktiven Statusmeldungen geben kann. Dies bedeutet, dass derzeit Befehle entgegengenommen werden können, darauf allerdings keine Rückmeldung erfolgt. Somit ist ein reeller Zustandaustausch in Echtzeit nicht möglich. Aus diesem Grund wurde nach jedem ausgehenden Befehl ein Intervall eingebaut, damit die Regel sequenziell abgearbeitet werden kann. Dafür wurden Zeitspannen definiert, die der Roboter in etwa braucht, um einen Ortswechsel, bzw. einen Sprachbefehl durchzuführen. Eine alternative Lösung, sofern der Service-Roboter Statusmeldungen geben könnte, wäre die feingranularere Aufteilung von einzelnen Prozessen, sodass erst nach erfolgreicher Rückmeldung des Roboters weitere Schritte eingeleitet werden würden.

Wird bspw. ein Sprachbefehl ausgeführt, so wird innerhalb der Regel über eine *Lambda*-Funktion der Zustandsraum erneut geändert, wodurch die inverse Transformation die Zustandsänderung überprüft und anhand deren der Sprachbefehl über ein MQTT-Topic an den Broker gesendet wird. Hierbei reagiert der Roboter auf das Topic und führt den Sprachbefehl aus. Diese Aktion ist dem Sequenzdiagramm (6.3) unter anderem in den Schritten 1.2:() -> zu entnehmen.

Der Anwendungsfall wurde erfolgreich umgesetzt und konnte mit dem realen Service-Roboter getestet werden. Ein *Mock*¹⁰ des Service-Roboters wurde entwickelt, um die Interaktionen zu simulieren und die Usability-Tests durchzuführen.

Notfall-Evakuierung mit einem Service-Roboter

Der Anwendungsfall der Notfall-Evakuierung enthält ähnliche Handlungsschritte als die des Check-ins, weswegen dieser nicht detailliert aufgegriffen wird. Die Prozessschritte sind ebenso möglich umzusetzen und wurden auch im Rahmen dieser Arbeit realisiert. Der Ablauf ist in einigen Aktionen zu vergleichen mit dem im Sequenzdiagramm 6.3 dargestellten.

Der Roboter selbst ist aktuell nicht in der Lage, Personen zu erkennen. Um dies zu vereinfachen, wurden im Rahmen dieser Arbeit lediglich bestimmte Orte angefahren und die Warnmeldung ausgeführt.

6.3 Fazit der prototypischen Implementierung

Die Anwendungsfälle konnten wie erwartet umgesetzt werden und zeigen, dass die konzeptionellen Entscheidungen eine Grundlage für das Framework darstellen. Die Regeln selbst können je nach Anwendungsfall beliebig definiert werden. An dieser Stelle werden dem Entwickler alle Möglichkeiten ohne Einschränkungen offengelassen. Des Weiteren ist der Schnitt zwischen Anwender und Framework klar gegeben, indem der Softwareentwickler sich ausschließlich mit den Regeln, Transformationen und Zustandsattributen befassen muss. Entscheidend ist die Gegebenheit eines MQTT-Brokers, ohne diesen das Framework keine Verbindung aufbaut und kein Datentransfer, an dem die Steuerzentrale teilnehmen kann, stattfindet.

¹⁰Simulierte Objekte in der Objekt-orientierten Programmierung. https://en.wikipedia.org/wiki/Mock_object
Besucht am 02.08.2022

Mit der Konzeption der Transformation wird bereits zu Anfang ein zentraler Punkt gegeben, mit dem der Anwender alle notwendigen Informationen bündeln und dem Frameworks zur Verfügung stellen kann. Anhand dieses Beschlusses kann der Entwickler die Zuordnung treffen, bspw. durch welches eingehende MQTT-Topic welches Attribut im Zustandsraum geändert werden soll. Mit der Entscheidung zur Nutzung der generischen Programmierung und somit der flexiblen Gestaltung des Zustandsraumes musste überlegt werden, wie dieses Objekt dennoch zu überwachen und zu manipulieren ist. Hierfür wurde die Java Reflection genutzt, die eine für diesen Fall geeignete Funktion der Java-Programmierung darstellt. Dennoch gibt es Hürden bei der Nutzung, die im Rahmen der Konzeption hingenommen wurden, die allerdings zu keinen Einschränkungen führen. Trotz dessen kann durch die Einhaltung der Modellierungsvorgaben (6.3.1) diese umgangen und so die Anwendungsfälle, die im Rahmen der Arbeit definiert wurden, bzw. in der Umgebung eines smarten Büros auftreten können, umgesetzt werden. Entscheidend dabei ist die Definition der Transformation sowie die der Regel selbst.

Der Prototyp erfüllt alle Voraussetzungen, um die definierten Anwendungsfälle umzusetzen.

6.3.1 Modellierungsvorgaben und -grenzen

Der Anwender muss für die Umsetzung jedes Anwendungsfalls folgende drei Punkte erneut implementieren:

1. *Das Attribut im Zustandsraum.* In der Klasse, in der die Zustandsattribute abgebildet werden, ist das Nutzen von einfachen Attributen, darunter *String*, *Integer*, *Double*, *boolean* und *Long*, empfohlen und vorgesehen. Es können zwar Objekte hinzugefügt werden, diese werden jedoch nicht vom Framework berücksichtigt.

Theoretische wäre die Nutzung von Komponentenobjekten möglich und umsetzbar, jedoch werden im Kontext dieser Arbeit ausschließlich die einfachen Attribute verwendet. Was wiederum zur Folge hat, dass durch die Nichtanwendung eine Grenze entsteht. Beim Versuch, den Zustandsraum um Objekte zu erweitern, wurde festgestellt, dass bei dem Kopiervorgang die Kopie eine neue Objektreferenz auf dem Feld der Komponente erzeugt. Dies hat eine ungewollte simulierte Modifikation der Komponente zur Folge, die bei jeder Zustandsänderung hervorgerufen wird, ohne dass sie tatsächlich geändert wurde und dadurch die inverse Transformation immer ausgeführt wird. Dies ist ebenso keine absolute Grenze, die nicht umgangen werden könnte, lediglich durch den zeitlichen Faktor nicht weiter ausgeführt wurde. Für die Behebung müsste gegebenenfalls ein optimierter Algorithmus zum Kopieren des Zustandsraums entwickelt, bzw. verwendet werden.

Da die Verwendung des Frameworks ausschließlich unter Gebrauch einfacher Attribute im Zustandsraum ohne weiteres funktioniert, wurde der Ansatz zur Nutzung von Komponenten vorerst verworfen.

2. *Die Transformation.* Die Transformation gibt ein klar definiertes Konstrukt vor. Dafür gibt es zwei Klassen, die implementiert werden können und jeweils die Oberklasse *AbstractTopicTransformation* erweitern. Zum einen ist dies die Transformation ohne optionale *White-List* von zugelassenen Werten und zum anderen diejenige, in der eine solche Liste übergeben werden kann. Über die *addTransformation*-Funktion wird zur Laufzeit das Transformationsobjekt erzeugt. Die Inhalte sind klar vorgegeben (siehe Abschnitt 6.2.4) und müssen vom Entwickler entsprechend definiert werden.
3. *Die Definition der Regel.* Die Definition einer Regel sieht vor, dass die Bedingung auf eine Zustandsänderung zutreffen muss. Der Regelprozess kann vom Anwender frei implementiert werden. Zum Erstellen einer Regel ist zu Anfang zu garantieren, dass diese von der abstrakten Regelklasse erbt. So kann die Regel einer Oberklasse zugeordnet und vom Framework als solche erkannt werden. Zusätzlich gibt es für die wichtigen Funktionen, die eine Regel zur Implementierung bereitstellt, Annotationen. Sie geben syntaktische Metadaten vor, die das richtige Implementieren der Funktionen forciert. Die Annotation muss beim Erstellen einer Klasse über der jeweiligen Funktion positioniert werden. Dem Quellcode-Ausschnitt im Anhang (siehe F.1) ist diese syntaktische Hilfestellung zu entnehmen. Wichtig ist ebenso das Übergeben der definierten Regel an das Framework mittels der *addRule*-Funktion.

Bei der Verwendung einer Bedingungsprüfung ist auf die Aussagekraft zu achten. Wird allerdings eine Bedingung so gewählt, dass sie bei jeder Zustandsänderung greift, kommt es zu einer Rückkopplung und endlosen Ausführung. Dafür muss entweder innerhalb der Regel der Zustand wieder geändert oder eine treffendere Bedingung definiert werden.

Diese Anmerkungen sind bei der Nutzung des Frameworks zu berücksichtigen, damit das Framework die Regeln und Prozesse entsprechend ausführt.

Kapitel 7

Evaluation

In diesem Abschnitt wird das erzielte Ergebnis des erarbeiteten Konzeptes (siehe Kapitel 5) und des Prototypen analysiert. Anhand von ausgewählten Forschungsmethoden werden die vorab identifizierten Anforderungen überprüft und evaluiert. Mithilfe der erhobenen Informationen wird zum Ende des Kapitels die Forschungsfrage sowie das Ziel der Arbeit abschließend betrachtet.

7.1 Usability-Test

Im Rahmen der Evaluation wurde ein Usability-Test durchgeführt, um bestimmte, für die Nutzbarkeit aufgestellte Anforderungen zu überprüfen. Das angestrebte Ziel und die Ergebnisse der Tests werden in dem folgenden Abschnitt erläutert:

7.1.1 Ziel des Usability-Tests

Anhand des Usability-Tests sind die Anforderungen zu prüfen, die den Bereich der Nutzbarkeit abdecken. Hierfür werden die Anforderungen, die den Tabellen (4.1, 4.2 und 4.3) zu entnehmen sind, betrachtet. Die dafür berücksichtigten Anforderungen sind konkret die Folgenden: NFA1, NFA2, NFA3 und NFA4 (siehe 4.2).

Damit die Erfüllung der Anforderungen überprüfbar ist, wird ausgewählten Anwendern der Zielgruppe eine Aufgabe gestellt, mit der das Framework getestet wird und dabei die oben genannten Anforderungen genauestens betrachtet werden. Ziel dabei ist die Überprüfung der Erfüllung der oben genannten Anforderungen an das System, bzw. Verbesserungen zu eruieren.

7.1.2 Durchführung des Usability-Tests

Für die Durchführung der Tests wurde vorab eine Aufgabe erstellt. Diese dient als Grundlage zur Identifizierung der Usability-Anforderungen. Der Aufbau und das Vorgehen der Testung wurde für jeden Teilnehmer identisch gestaltet. Vor Beginn wurde der Kontext sowie das Framework und dessen Kernkomponenten, die Rahmenbedingungen und die Aufgabenstellung selbst erläutert. Die Aufgabe war so zu erstellen, dass sie umgebungsunabhängig erfüllt werden konnte. Da die Tests ausschließlich auf die Nutzung des Frameworks abzielten, war die Umgebung sowie die Anbindung

verfügbarer Geräte zu vernachlässigen. Voraussetzung war die Verfügbarkeit eines MQTT-Brokers. Durch die fehlende Anbindung von reellen Geräten wurden die ausgehenden MQTT-Nachrichten durch eine Simulation, konkret die eines Service-Roboters, verarbeitet. Eingehende Nachrichten wurden simuliert, indem MQTT-Nachrichten manuell veröffentlicht wurden.

Die von den Anwendern durchzuführende Aufgabe ist wie folgt definiert:

Es soll ein Anwendungsfall implementiert werden, bei dem ein Service-Roboter einen Mitarbeiter oder Gast, der an der Tür steht, empfangen und begrüßen soll. Die Eingangsbedingung ist die Authentifizierung über eine Kamera. Der ganze Vorgang wird simuliert, indem die MQTT-Nachricht manuell erzeugt und ausgelöst wird. Auch die Definition des MQTT-Topics ist vorgegeben. Danach soll über die Steuerzentrale eine Regel ausgeführt werden, anhand derer der simulierte Service-Roboter gesteuert wird. Dafür sind folgende Anforderungen zu erfüllen:

- Nach eingehendem Topic soll der Service-Roboter angesteuert und an die Tür geschickt werden. (Die Ansteuerung des Service-Roboters erfolgt ebenso über MQTT. Da in den meisten Fällen kein Roboter verfügbar ist, wird auch diese Kommunikation mittels MQTT simuliert.)
- Ist der Roboter an der Tür, soll er die Begrüßung starten. (Die folgende Interaktion wird im Rahmen des Test ebenso mittels der Simulation durchgeführt. Wird die gesendete Nachricht über den Service-Roboter-*Mock* auf der Konsole ausgegeben, so gilt die Aufgabe als erledigt.)

Für die Aufgabe sind folgende Punkte zu erfüllen:

- Die Kenntnis über MQTT-Topics, die für die Kommunikation benötigt werden.
- Ein Zustandsraum, der alle benötigten Komponenten abbildet.
- Der Service-Roboter als Komponente, sodass dieser bei Ausführung einer Aufgabe für weitere Aufgaben gesperrt werden kann.
- Der Auslöser, welcher durch ein MQTT (PlugIn) abgebildet wird.
- Die Transformation der eingehenden MQTT-Topics zu Zustandsänderungen, auf die Regeln ausgeführt werden sollen.
- Die Regel, die den Vorgang der Begrüßung und des Check-ins vorgibt.

Dem Anhang (siehe G) ist das Dokument zur Durchführung des Usability-Test beigefügt.

Nach dem Abschluss der Aufgabe wurden die Probanden um ihre Meinung mithilfe eines Fragebogens, der sich an dem *System Usability Scale (SUS)* Template orientiert, gebeten. Dieser Fragebogen ist ebenso dem Anhang (siehe G) zu entnehmen. Auf die Auswertung wird in der Zusammenfassung (7.1.3) des Usability-Tests eingegangen.

7.1.3 Fazit

Der Usability-Test konnte erfolgreich durchgeführt werden. Das Ergebnis der quantitativen Forschung kann nicht als repräsentativ eingestuft werden, da es sich um eine kleine Auswahl an Experten handelt. Es zeigt dennoch eine Tendenz, welche Anforderungen abgedeckt sind, bzw. welche Verbesserungspotentiale in dem Framework stecken. Das Resultat der Aufgabe ähnelte sich bei den Probanden mit Ausnahme der Namensgebung der Attribute im Zustandsraum.

Folgender Auflistung ist der durchschnittliche Wert der Antworten zu entnehmen:

1. *Ich denke, dass ich dieses System gerne öfter nutzen würde. (80 Punkte)*
2. *Ich fand das System unnötig komplex. (8 Punkte)*
3. *Ich fand das System einfach zu bedienen. (76 Punkte)*
4. *Ich denke, dass ich die Unterstützung einer technischen Person benötigen würde, um dieses System nutzen zu können. (9 Punkte)*
5. *Ich fand, dass die verschiedenen Funktionen in diesem System gut integriert waren. (86 Punkte)*
6. *Ich dachte, es gäbe zu viele Inkonsistenzen in diesem System. (0 Punkte)*
7. *Ich könnte mir vorstellen, dass die meisten Leute sehr schnell lernen würden, dieses System zu benutzen. (89 Punkte)*
8. *Ich fand das System sehr umständlich zu bedienen. (5 Punkte)*
9. *Ich fühlte mich sehr sicher mit dem System. (76 Punkte)*
10. *Ich musste viele Dinge lernen, bevor ich mit diesem System loslegen konnte. (19 Punkte)*

Im Rahmen der Bewertung, bei der 100 die volle Zustimmung und 0 die Ablehnung darstellt, ist das Ergebnis sehr positiv und zeigt das Potential des Frameworks.

Nachdem der Test abgeschlossen und der Fragebogen beantwortet war, wurde die befragte Person abschließend noch interviewt, damit Eindrücke über den Test hinaus gesammelt werden konnten. Diese Interviews werden im folgenden Abschnitt resümiert.

7.2 Experteninterview

Zur umfangreichen Informationsgewinnung wurde anschließend zu dem Usability-Test ein Interview mit dem jeweiligen Probanden durchgeführt. Hierbei wurden sowohl die gesammelten Erkenntnisse und Erfahrungen während der Aufgabe erfragt sowie auf die Anforderungen (siehe Kapitel 4.5) eingegangen. Ein Teil der Probanden wurde dafür bereits bei der Erhebung von Anforderungen befragt. Somit konnte sich ein Bild verschaffen werden, wie die Anforderungen umgesetzt, bzw. diese von den Experten empfunden wurden.

7.2.1 Ziele des Experteninterviews

Das Ziel des abschließenden Experteninterviews ist es, die Erfahrungswerte der Probanden zu sammeln, um diese den Anforderungen gegenüberzustellen. Bei Experten, die bereits zu der Anforderungserhebung interviewt wurden, konnte konkret auf die Umsetzung der Kriterien und Anforderungen eingegangen werden. Ebenso ist ein abschließendes Feedback wünschenswert, sowie ein Identifizieren der Herausforderungen für den Anwender und der eventuellen Schwachstellen des Systems.

7.2.2 Fazit

Das Experteninterview wurde, vergleichbar zu dem vorherigen Interview (siehe 1.4.1), mit einem semi-strukturierten Ansatz durchgeführt. Anfangs wurde an den Fragebogen angeknüpft, um eine nachträgliche Zusammenfassung der Erkenntnisse jedes einzelnen zu erfahren. Sofern die Zusammenfassung bereits die notwendigen Informationen enthielt, wurde ein unstrukturierter Ansatz verfolgt und ein offener Dialog geführt. War die Zusammenfassung jedoch nicht aufschlussreich, so wurden anschließend konkret die folgenden Fragen gestellt:

1. „Wie ist Ihr erster Eindruck des Systems?“
2. „Gibt es aus Ihrer Sicht (Sicht des Anwenders) Mängel, die Ihnen die Nutzung des Frameworks erschweren?“
3. „Gab es Unklarheiten während der Anwendung des Frameworks? - Wenn ja, welche?“
4. „Haben Sie eine Idee oder Lösung, um diese Unklarheit zu beseitigen?“
5. „Gibt es allgemein Anregungen, bzw. Vorschläge für Verbesserungen?“
6. „Was gefällt Ihnen an dem Framework am meisten? - Was gefällt Ihnen nicht?“

Rekapitulierend wurde ein aufschlussreiches und erfreuliches Ergebnis erzielt. Die Antworten gaben ein klares Meinungsbild von der Einfachheit der Handhabung der formalisierten Interaktionen der Steuerzentrale wieder und bestätigen die Erfüllung des Ziels dieser Arbeit. Aus Sicht des Anwenders gab es wenige Anmerkungen zum Aufbau des Frameworks.

Eine erwähnenswerte Anmerkung war folgende:

„Wenn man wenig bis keine Erfahrung mit der Java Entwicklung hat, gibt es klar die ein oder andere Wissenslücke, die zuerst beseitigt werden muss. Durch die wenigen Handlungsschritte, die jedoch viel Interpretationsfreiheiten bieten, besonders bei der Regeldefinition, ist dies doch überschaubar. Bekommt man dementsprechend eine Hilfestellung, bzw. eine detailliertes Beispiel an die Hand, so können kleinere Schritte und Anwendungsfälle relativ schnell umgesetzt werden.“

Ein Ergänzungs-, bzw. Verbesserungsvorschlag, der während eines Interviews erwähnt wurde, ist die Optimierung des Anlegens von sämtlichen Objekten. Hierfür wurde vorgeschlagen, eine Art

Command Line Interface (CLI), wie bspw. die von Angular¹, zu implementieren. Die Idee war Befehle über die Kommandozeile geben zu können, bspw. das Anlegen von Attributen im Zustandsobjekt, das Erzeugen leerer, bzw. mit den dafür vorgesehenen Parameterwerten befüllter Transformationen, und das Erstellen eines leeren Regelkonstruktes. Dadurch würde der notwendige Code generiert und dem Entwickler zur Verfügung gestellt werden, wodurch dieser sich ausschließlich um das Implementieren der Regeln kümmern muss.

Abschließend lässt sich anmerken, dass mit der konkreten Anweisung, welcher Anwendungsfall umgesetzt werden soll, bzw. welche Randbedingungen dafür gelten, schnell ein Ergebnis erzielt werden kann.

7.3 Evaluation der Anforderungen

Im Rahmen der Arbeit werden die Anforderungen (siehe Kapitel 4.5) zum Abschluss anhand von dafür vorgesehenen Praktiken evaluiert. Die Reihenfolge entspricht der tabellarischen Aufstellung der Anforderungen. Zuerst erfolgt die Abarbeitung der *funktionalen Anforderungen*, anschließend die der *nicht funktionalen Anforderungen* und zum Ende hin die der *zusätzlichen Anforderungen*.

Funktionale Anforderungen

Während der Konzeption und Entwicklung des Frameworks wurden die funktionalen Anforderungen (FA1 - FA6, siehe Tabelle 4.1) berücksichtigt. Sie stellen den wichtigsten Kernpunkt dar. Deren Erfüllung konnte durch die abschließenden Experteninterviews identifiziert werden. Die Resonanz der Experten gab auch deutlich wieder, dass auf Grundlage der Anforderungen der Prototyp aufbaut. Die Struktur des Regelobjektes (FA1) wird gegeben, indem eine Klasse von der Oberklasse erbt und diese erweitert. Umgesetzt wurde diese Anforderung mithilfe des *Template Method* Architekturnusters, auf das auch bereits in der Konzeption (siehe Kapitel 5) eingegangen wurde.

Die Anforderung (FA2) ist umgesetzt, indem der Regel mit der *ruleTrigger*-Methode ein Wert eines *Enum Types*² zugeordnet wird. Ein essentieller Teil des Frameworks ist der vom Anwender zu implementierende Zustandsraum, welcher über eine separate Klasse vorgegeben wird. Dadurch ist auch die Anforderung (FA3) als umgesetzt anzusehen. Die Individualität der Implementierung des Zustandsraume ist durch die frameworkseitige Verwendung der Java *Generics* gegeben. Demzufolge arbeitet das Framework mit dem vom Entwickler übergebenen Klassenobjekt.

Die Ausprägungen der Bedingungen zur Ausführung der Regelprozesse sind unverzichtbar. Sie müssen vom Anwender aufgestellt werden (FA4). Damit eine Aufstellung erfolgen kann, ist in der Regelklasse eine Funktion gegeben, die das Implementieren der Bedingung erzwingt. Die darauffolgende Forderung (FA5) ist prinzipiell gegeben, wird allerdings durch die Einschränkung im

¹TypeScript basiertes Web Application Framework. <https://angular.io/> Besucht am 05.08.2022

²Spezieller Datentyp. <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html> Besucht am 05.08.2022

Zustandsraum aktuell nicht als geeignet angesehen. Die Nutzung von Java *Reflection*, bzw. die Umsetzung der inversen Transformation erschwert dies. Derzeit wird von der Komponenteneinbindung in das Zustandsobjekt abgeraten. Das Erstellen von Komponenten ist dennoch möglich und kann im Regelkontext angewendet werden, sofern bestimmte Eigenschaften für eine Regel relevant sind.

Der letzte Punkt (FA6) ist ebenso umgesetzt, da innerhalb der Regel bis auf die Voraussetzung der Implementierung der Regelbedingung sowie des -prozesses keine Einschränkungen bestehen. Zusätzlich können bspw. sämtliche HTTP-Abfragen getätigt werden. Im Falle des Use Cases Kheck-in mit einem Service-Roboter" werden HTTP-Abfragen ausgeführt, damit Informationen der Büroplatzbuchungssoftware abgerufen werden können.

Nicht funktionale Anforderungen

Zur Überprüfung der *nicht funktionalen Anforderungen* wurden Methoden für deren Überprüfung verwendet. Zusätzlich zu den verifizierten *User Stories* mittels der Usability-Tests konnten ebenso die aufgestellten Forderungen, die allgemein die Benutzerfreundlichkeit (BF) umfassen, überprüft werden. Der durch die Experteninterviews aufgestellte Anspruch (NFA1) konnte durch die Usability-Tests erfüllt werden, da die Probanden selten mehr als eine Aktion benötigten, um die bereits erstellte Regel dem Framework zu übergeben. Grund dafür ist die Funktion der Steuerzentrale *addRule(...)*. Der Anwender kann die Aufgabe der Funktion schnell zuordnen und bekommt über die Entwicklungsumgebung die erwarteten Übergabeparameter mitgeteilt.

Im Rahmen des Usability-Tests waren alle Teilnehmer in der Lage eine Regel zu erstellen und diese dem Regelwerk hinzuzufügen (NFA2). Ausgehend von dieser Anforderung wurden während der Experteninterviews Anregungen angebracht, die gegebenenfalls eine weitere Vereinfachung für den Entwickler darstellen. Diese werden im Ausblick (siehe Kapitel 9) aufgegriffen.

Durch die rein programmatische Anwendung des Frameworks kann der Entwickler zu jedem Zeitpunkt auf die Funktionen der Steuerzentrale zugreifen (NFA3). Dies wurde ebenso von den Testteilnehmern im anschließenden Experteninterview bestätigt. Zum aktuellen Zeitpunkt sind die nutzbaren Funktionen sehr überschaubar und decken die notwendigsten Funktion ab, darunter das Hinzufügen von Regeln, Erstellen aller wesentlichen Objekte und Komponenten, Aufsetzen der erforderlichen MQTT-Konfiguration, Ergänzen von Transformationen, Starten des MQTT-Clients und das Erweitern um zeitbasierte Regelauslöser.

Die Anforderung (NFA4) ist ebenso gegeben, da der Anwender sich ausschließlich um die Definitionen der Regeln, der Transformationen und des Zustandsraumes kümmern muss. Dennoch gibt es zur weiteren Vereinfachung der Regeldefinition zusätzliche Möglichkeiten, die im Rahmen dieser Arbeit nicht umgesetzt wurden. Diese werden im Ausblick aufgegriffen, um weitere Potentiale aufzuzeigen.

Ein nennenswerter Aspekt ist die allgemeine Zuverlässigkeit des Systems. Zur Überprüfung der Zuverlässigkeitssanforderung (NFA5) wurden im Rahmen der Entwicklung *JUnit-Tests*³ implementiert,

³Test Framework für Java. <https://junit.org/junit5/> Besucht am 08.08.2022

die einen bestimmten Sachverhalt auslösen und überprüfen. Demnach wird ein Zustand erwartet, der nach dem Auslösen einer Regel eintreffen sollte. Anschließend wird der Ist-Zustand mit dem Soll-Zustand abgeglichen und somit die Zuverlässigkeit überprüft. Dennoch gibt es eine starke Abhängigkeit zur Regelbedingung, die der Anwender definiert. Ist diese nicht deutlich genug, kann es passieren, dass Regeln ungewollt ausgeführt werden. Deshalb ist die Validierung der Regelbedingung durch den Entwickler essentiell. Somit ist die Zuverlässigkeit abhängig von den Interaktionen und der Implementierung des Anwenders. Grundsätzlich gibt es systemseitig keine negativen Auswirkungen von mehrdeutigen Bedingungen, die fälschlicherweise die Zuordnung und Ausführung einer Regel gefährden. Wird jedoch durch die Regel der Zustand an einer anderen Stelle erneut geändert, ohne dass sich der vorherige Wert und dessen Bedingung wieder negiert hat, so würde das Framework eine Endlosschleife und Rückkopplung bilden und dadurch einen *StackOverflowError* erzeugen. Demnach ist die Auswahl der Bedingung einer der wichtigsten Punkte, die zu berücksichtigen sind.

Durch die Verwendung von *Threads* werden Regeln, deren Bedingungen im Vorfeld zutreffen, asynchron ausgeführt. Es können dadurch voneinander unabhängige Regel parallel abgearbeitet werden. Somit ist die Performanz des Systems erhöht (NFA6). Die Reaktionszeit und die Performanz der Kommunikation ist sowohl abhängig von den jeweiligen Protokollen und Technologien, die eingesetzt werden, sowie von der dafür verwendeten Hardware und Infrastruktur (NFA7). Eine konkrete Überprüfung der Anforderungen wurde im Rahmen der Arbeit nicht durchgeführt, da diese zum aktuellen Zeitpunkt keine Auswirkungen auf das System selbst haben. Dennoch ist unter Verwendung der derzeitigen Technologien eine gewisse Performanz vorauszusetzen und zu erwarten.

Die *nicht funktionalen Anforderungen*, die allgemein die Verfügbarkeit des Systems beschreiben (NFA8 & NFA9), sind aktuell zu vernachlässigen und wurden im Rahmen dieser Arbeit nicht überprüft. Das Framework selbst produziert von sich aus keine Downtime. Entscheidend sind dabei die Richtigkeit der Regeldefinition, Verbindungen zum MQTT-Broker und Internet, sowie die Stromversorgung, da das Framework lokal auf einem *Raspberry Pi* betrieben wird.

Zur Überprüfung der Fehlertoleranz (NFA10) wurden die *JUnit-Tests* um weitere Testfälle ergänzt. Hierbei wurde eine fehlerhafte Regel definiert, die innerhalb des Tests ausgelöst und ausgeführt wurde. Die Resultate erzielten eine gewisse Fehlertoleranz, die durch eine Weiterentwicklung stets verbessert werden kann.

Der letzte Punkt (NFA11) der Tabelle (4.2) zählt als Erweiterung des Frameworks, sodass ein ständiger Informationsaustausch stattfindet und der Entwickler das System überwachen kann und zu jedem Zeitpunkt einen Einblick in die Prozesse erlangt. Die Integration solch einer Lösung in das System ist ohne weiteres möglich.

Zusätzliche Anforderungen (Randbedingungen)

Die Tabelle (4.3) beinhaltet weitere Anforderungen und Randbedingungen, die mithilfe der verwendeten Erhebungstechniken identifiziert wurden.

Die Randbedingung (ZAF1), Nutzung der Programmiersprache Java, wurde eingehalten. Das Fra-

mework ist ausschließlich in Java entwickelt.

Für den Zustandsraum ist eine gewissen Modellierungsempfehlung vorgegeben, die keine Objekte als Felder im Zustandsraum zulässt. Die Abbildung von reellen Zuständen findet ausschließlich über Attribute statt. Somit ist die Mindestanforderung für (ZAF2) erfüllt.

Die Anforderung (ZAF3) ist abgedeckt, da die Regelprüfung uns -ausführung über einen *Thread Pool* in einen eigenen *Thread* ausgelagert wird. Ebenso ist (ZAF4) erfüllt, da die Kommunikation zum aktuellen Zeitpunkt ausschließlich über MQTT erfolgt. Das Auslösen von Regeln über definierte Zeitpunkte kann durch *Cronjobs* oder durch die Annotation (*Scheduled*) des Spring Frameworks stattfinden. Demnach stellen diese beide Möglichkeiten die einzigen Optionen eines Auslösers dar. In Folge dessen gilt auch die Forderung (ZAF5) als erledigt.

Durch die Verwendung der generischen Programmierung in Java ist sichergestellt, dass das Zustandsobjekt zur Laufzeit dem Framework übergeben wird (ZAF6). Somit kann das Framework mit dem Objekt arbeiten, unabhängig von der Struktur der Klasse.

Anhand der Anforderung (ZAF7) findet die Kommunikation überwiegend über MQTT statt. Ein Punkt, der nicht vom Framework abgedeckt ist, ist die Sicherheit der Kommunikation. Der Anwender muss sicherstellen, dass dementsprechende Vorkehrungen getroffen sind, damit der MQTT-Broker und die darüber laufende Kommunikation abgeschirmt und für Dritte nicht zugänglich sind (ZAF8). Im Rahmen der Arbeit ist der MQTT-Broker ausschließlich im lokalen Netzwerk verfügbar und erlaubt den Zugriff nur für bestimmte Nutzer und Kommunikationsteilnehmer.

Der *Single Point of Contact* ist gegeben, da die Einstellungen und das Hinzufügen von Regeln und Transformationen ausschließlich über eine einzige Klasse funktionieren, die *InnovationLogicHubApplication*.

Nachdem die aufgestellten Anforderungen evaluiert wurden, wird im folgenden Abschnitt auf die Evaluation der Forschungsfrage eingegangen.

7.4 Evaluation der Forschungsfrage

Der in dieser Arbeit verfolgte Ansatz bietet dem Softwareentwickler eine Möglichkeit, mit wenigen Schritten eine Prozessautomatisierung oder Regeldefinition basierend auf einem Anwendungsfall im Bereich des intelligenten Büros, bzw. *Smart Office* oder *Smart Home*, umzusetzen. Hierbei wird im Vergleich zu anderen Lösungen, darunter openHAB und Home Assistant, nicht der Weg über eine Integration angestrebt. Die Gestaltung solcher Komponenten, die über Integrationen abgebildet werden, findet in dem konzipierten Framework über den Zustandsraum statt. Er lässt dem Entwickler die Ausprägung offen. Zusätzlich wird darauf verzichtet eine Benutzeroberfläche bereitzustellen, mit der die Interaktionen gegebenenfalls aufwändig, bzw. mehrschrittig sein könnten. Durch die alleinige Nutzung der Programmiersprache Java ist innerhalb des Frameworks kein Kontextwechsel, bzw. Syntaxwechsel erforderlich, wodurch die Interaktionen ebenso vereinfacht bleiben. Mithilfe der reduzierten Interaktionspunkte für den Entwickler ist gleichermaßen eine Formalisierung der Interaktionen gewährleistet. So ist lediglich pro Anwendungsfall eine klar vorgegebene Struktur zu implementieren. Diese besteht aus mindestens einer Transformation, einer Regel und einem Attribut im Zustandsraum.

Die Usability-Tests sowie die Experteninterviews zeigen das Potential dieses Systems, welches bei Weitem nicht ausgeschöpft ist und in Zukunft vorangetrieben werden kann.

Durch die einfache Erweiterbarkeit, die durch die Architektur gewährleistet ist, können nützliche Funktionen hinzugefügt werden.

In der Gesamtheit wurde das System als überaus hilfreich und Entwicklungsfähig eingestuft. In dieser Arbeit stehen ausschließlich die Konzeption, Grundlagenschaffung und prototypische Entwicklung zur Erreichung der Zielsetzung im Fokus. Die Grundlagen und -funktionen wurden implementiert und zur Verfügung gestellt.

Das Framework kann dennoch nicht direkt mit den bestehenden Softwarelösungen, bspw. Home Assistant und openHAB, verglichen werden, da diese weitaus mehr Funktionalitäten anbieten, die im Rahmen dieser Arbeit nicht vorgesehen waren.

Zusammenfassend lässt sich sagen, dass die Usability von Prozessautomatisierungen und Regeldefinitionen durch das Framework optimiert werden können. Dieses bietet klare Strukturen und zeigt alle Handlungsschritte auf. Die Softwareentwickler erfahren durch das Framework die einfache Handhabung der formalisierten Interaktionen. Das System bietet einen möglichen Lösungsweg, der die Forschungsfrage beantwortet.

Kapitel 8

Fazit

Ziel dieser Arbeit war die Konzeption und prototypische Umsetzung einer Steuerzentrale für ein smartes Büro mit dem Fokus einer einfachen Handhabung der formalisierten Interaktionen für Softwareentwickler. Das in diesem Rahmen entstandene Framework bietet eine klare Struktur zur Implementierung von Anwendungsfällen, die der Anwender im Umfeld eines smarten Büros umsetzen möchte. Durch den rein programmatischen Ansatz ist somit dem Softwareentwickler ein System gegeben, das ausschließlich mit Java bedient wird. Verwendete Entwurfsmuster schaffen eine wartbare, erweiterbare und modulare Architektur. Durch die vorgegebene Schablone der Regel ist dem Entwickler zwar eine Struktur vorgeschrieben, die dennoch in der Ausprägung des jeweiligen Anwendungsfalls alle Handlungsmöglichkeiten und Freiheiten offen lässt.

Für die Erreichung des Ziels wurden unter anderem Recherchen und Analysen durchgeführt, die den Stand der Technik hervorbrachten, gefolgt von der Testung derzeitiger Softwarelösungen, um deren Möglichkeiten und Grenzen zu erfahren. Auf dieser Grundlage wurden weitere Schritte eingeleitet, damit gezielte Anforderungen definiert werden konnten. Hierfür wurde eine Markt- sowie Zielgruppenanalyse vorgenommen und konkrete Anwendungsfälle definiert. Mit diesen Hintergründen wurden zur Auffassung möglichst vieler Anforderungen zusätzlich Experteninterviews durchgeführt, wodurch die Konzeption und Umsetzung realisiert werden konnte.

Unter Berücksichtigung der zu gewährleistenden Modularisierung, Erweiterbarkeit und Wartbarkeit wurden dementsprechende Architekturentwurfsmuster verwendet. Zudem wurden einzelne Schichten des Systems entworfen, darunter zählt die Kommunikations- sowie Logik- und Persistenzschicht, wobei letztere nicht konkret konzipiert wurde.

Ein konkretes Konstrukt für die Regeldefinition wurde verwendet, damit der Anwender eine klare Struktur vorfindet und so die einfache Handhabung der formalisierten Interaktionen gefördert wird. Darüber hinaus wurde eine Annotation ergänzt, die eine Regel als solche deutlich markiert.

Zur Überprüfung und Bestätigung der Anforderungen sowie des gestellten Ziels wurden anschließend für die Umsetzung Usability-Tests und zusätzliche Experteninterviews durchgeführt. Diese halfen dabei, den erarbeiteten Prototypen und dessen Konzept zu evaluieren. Alle priorisierten Anforde-

rungen konnten erfüllt werden. Das Ergebnis der Usability-Tests sowie der Interviews ergaben ein positives Feedback. Dennoch gibt es Verbesserungs-, bzw. Anpassungspotentiale, die im Folgenden thematisiert werden. Erweiterungen und mögliche Weiterentwicklungen werden ebenso im Ausblick (siehe Kapitel 9) aufgegriffen.

Zum aktuellen Zeitpunkt ist die Verwendung von Komponenten nicht vorgesehen, da diese im Zustandsraum derzeit keine Verwendung haben. Die Gründe dafür sind unter anderem die Entscheidung der Verwendung der Java Reflection sowie die aktuelle inverse Transformation, die jedoch in weiteren Iterationen angepasst werden kann. Zusätzlich gibt es mögliche Optimierungen der Regelausführung, bzw. deren Management, um die Performanz zu steigern und eine strukturierte Abfolge zu gewährleisten.

Dennoch liefert das Konzept sowie der aktuelle Prototyp alle Grundvoraussetzungen, um das gewünschte Ziel zu erreichen, bzw. Prozesse im intelligenten Büro zu automatisieren. Unter Berücksichtigung der Modellierungsgrenzen, als auch der -vorgaben ist die Realisierung von komplexen Anwendungsfällen, bspw. die Verwendung eines Service-Roboters, ohne weiteres möglich.

Das Ziel dieser Master-Thesis, eine Steuerzentrale für ein intelligentes Büro zu konzipieren und entwickeln, welches den Fokus der einfachen Handhabung der formalisierten Interaktionen für Softwareentwickler besitzt, wurde demnach erreicht: das hier erarbeitete und vorgestellte Framework optimiert die Usability in Form der Programmierung und ermöglicht das Umsetzen und Ausführen von Regeln.

Kapitel 9

Ausblick

Das Framework liefert aus Sicht der Usability ausgezeichnete Ergebnisse und gibt dem Anwender die zu implementierenden Elemente klar vor. Vorausgesetzt werden hierbei jedoch Grundkenntnisse der Programmierung, sowie die Kenntnis über die zu implementierenden Elemente des Systems. Zukünftige Weiterentwicklungen des im Rahmen dieser Arbeit vorgestellten Frameworks sollten demnach zum Ziel haben, das entworfene Konzept sowie den aktuellen Prototypen entsprechend zu erweitern.

Für die Optimierung des Systems bietet primär der Algorithmus der Regelprüfung und -ausführung Potential. Dieser ließe sich beispielsweise um einen *Queuing*-Mechanismus ergänzen, damit alle eintreffenden Zustandsänderungen berücksichtigt und nacheinander abgearbeitet werden. Kann eine Aktion auf Grundlage der Änderung nicht durchgeführt werden, würde sie zurückgehalten und die Bedingung immer wieder überprüft werden, bis diese zutrifft und die Zustandsänderung eine entsprechende Aktion ausführt, sofern dies beabsichtigt ist.

Des weiteren kann das derzeit statische *JSON*-Konstrukt der inversen Transformation so optimiert werden, dass beliebige Konstrukte als MQTT-Nutzlast veröffentlicht werden können. Dadurch wäre eine Versendung mehrerer Informationen möglich.

Denkbar wäre eine Optimierung des Kopieralgorithmus, sodass verwendete Komponenten im Zustandsraum nach dem Kopiervorgang keine neue Objektreferenz zur Programmlaufzeit erhalten. In Kombination dieser Schritte wäre nachfolgend eine Verwendung von Komponenten im Zustandsraum denkbar, wobei dies nach wie vor keine konkreten Vorteile aufweist. Dadurch würde lediglich die Gestaltung des Zustandsraumes auf die Komponenten verlagert werden. Die Java Reflection müsste dahingehend geringfügig angepasst werden, sodass Überprüfungen im Stile des absteigenden Baumgraphs entlang der Pfade stattfindet.

Neben den inhaltlichen Optimierungen gibt es ebenso Erweiterungsmöglichkeiten, um die formalisierten Interaktionen der Softwareentwickler zusätzlich zu vereinfachen und weitere Funktionalitäten dem Framework zu übergeben. Damit der Softwareentwickler zu den Annotationen und der vorgegebenen Regelstruktur dessen Richtigkeit überprüfen kann, wäre ein Debug-Mechanismus

für die implementierten Regeln sinnvoll. Mithilfe diesem könnte zusätzlich die Syntax und zu verwendende Struktur der Regel sichergestellt werden. Infolgedessen wäre ein Ansatz für ein Testframework zu konzipieren, das ergänzend zu JUnit-Tests die Funktionalitäten des Systems überprüft.

Eine weitere Maßnahme, mit der die formalisierten Interaktionen der Softwareentwickler künftig noch einfacher zu handhaben sind, wäre die Entwicklung eines CLIs. Dadurch könnten mithilfe dynamischer Quellcode-Generierungen die einzelnen Komponenten, darunter Attribute im Zustandsraum, leere und gefüllte Transformationsobjekte und Regelklassen, erstellt werden. Als Grundlage dafür würde beispielsweise das CLI von *Apache Maven*¹ dienen. Über einen einfachen Befehl könnten alle Elemente, die für eine Regel notwendig sind, erstellt werden. Es entfiele z.B. die eigenständige Erstellung der Regel. Wobei derzeit zu beachten ist, dass die richtige abstrakte Klasse verwendet wird und alle zu implementierenden Funktionen präsent sind.

Alternativ wäre die Entwicklung eines Integrated Development Environment (IDE) Plug-ins möglich, anhand dessen das Regelkonstrukt erzeugt werden kann.

Zum stetigen Ausbau des Frameworks wäre eine Erweiterung zur Nutzung zusätzlicher Kommunikationsprotokolle durchaus denkbar. Die notwendigen Vorkehrungen sind durch die Architektur gegeben. Ähnlich zu Home Assistant wäre die Bereitstellung einer MQTT-Broker-Instanz möglich, sodass das Framework autark und unabhängiger verwendet werden kann, bzw. das Management der grundlegenden Funktionen gebündelt wird.

Zur Steigerung der Integrität und Konsistenz des Zustandsraumes kann mittels der Persistenzschicht beispielsweise eine Transaktionshistorie erstellt werden, die jede Zustandsänderung als Datenmodell, bzw. Objekt in eine Datenbank speichert. So kann nach einem Ausfall der konkrete Zustandsraum wieder eingespielt und die eigentlichen Zustände verwendet werden. Aktuell müssten alle Geräte ihren derzeitigen Zustand erneut übermitteln, wodurch die Steuerzentrale diese nochmals registrieren könnte. Durch das Einspielen einer Transaktion würden alle Zustände im Zustandsraum wieder vorhanden sein. Die Ausprägung dieser Schicht ist für weitere Entwicklungen offen gehalten.

Für die Fähigkeit der Überwachung von aktuell ablaufenden Regeln könnte ein sogenanntes *Monitoring* entwickelt und beliebig ausgeprägt werden, sodass auch Prognosen erhoben und Interaktionspunkte geschaffen werden könnten. Über einen separaten Monitor könnte beispielsweise im Büro angezeigt werden, welche Regel durchlaufen wird, bzw. wie die Zustände im Zustandsraum aussehen.

Der Ansatz bietet allgemein eine Möglichkeit, in einer Umgebung neue Automationen und Prozesse schnellstmöglich umzusetzen. Durch das stetige Wachstum und die Akzeptanz des Bereiches Smart Home und dessen Spezifikation des *Smart Office* wäre das Vorantreiben dieser Idee durch ein Open-Source-Ansatz möglich. Die geschaffene Grundlage ermöglicht die Umsetzung aller Optimierungs- sowie Erweiterungsmöglichkeiten.

¹Kommandozeilenwerkzeug aus der Kategorie der Build-Werkzeuge. <https://maven.apache.org/index.html>
Besucht am 12.08.2022

Abbildungsverzeichnis

2.1	Technologische Einordnung von IoT [SIEPERMANN und LACKES 2018]	10
2.2	Exemplarische Darstellung eines IoT-Systems [GILLIS 2022]	11
2.3	Client-Server, Peer-to-Peer und Message Passing Interaktionsparadigma [MINERVA, BIRU und ROTONDI 2015]	13
2.4	Mögliche Anwendungsszenarien im Smart Home [STRESE u. a. 2010]	18
2.5	Technologische Einordnung von Smart Home in Verbindung zu IoT [BENDEL 2021] .	19
2.6	Aufbau und Funktionsweise einer Smart Home Infrastruktur	20
2.7	Themenbasiertes Pub/Sub Kommunikationsmodell [HUNKELER, TRUONG und STANFORD-CLARK 2008]	26
2.8	Architektur des Home Assistant Supervisors [SCHOUTSEN 2020]	30
2.9	Architektur des Home Assistant Cores [LI u. a. 2018]	31
2.10	Architektur der openHAB Plattform [KREUZER 2013]	37
2.11	Architektur der openHAB 2.0 Plattform [KREUZER 2016]	38
2.12	OSGi Schichtenarchitektur [KREUZER 2022]	39
3.1	Strategie der Literatursuche	43
4.1	Übersicht der repräsentativen Schlüsselanbieter [LASQUETY-REYES 2021]	52
4.2	Umsatz-Prognose von Deutschland und den USA [LASQUETY-REYES 2021]	53
4.3	Globaler Smart Home Marktwert [LASQUETY-REYES 2021]	53
4.4	Anzahl Smart Home nach Segment [LASQUETY-REYES 2021]	56
4.5	Persona zur Zielgruppenanalyse	58
4.6	Use Case 1 - Anwendungsfalldiagramm	60
4.7	Use Case 2 - Anwendungsfalldiagramm	63
5.1	Infrastruktur des Anwendungsumfeldes der Steuerzentrale	71
5.2	Grober Programmablauf des Frameworks	75
5.3	Schichtenarchitektur	77
5.4	Architektur mit verwendeten Entwurfsmustern	80
5.5	Klassendiagramm der Kommunikationsschicht	81
5.6	Sequenzdiagramm zur Kommunikationsschicht	81
5.7	Klassendiagramm zur Logik- und Prozessschicht	82
5.8	Sequenzdiagramm zur Logikschicht	83

6.1	Transformation über Switch-Case-Anweisung	90
6.2	Transformation über eigenes Objekt	92
6.3	Check-in Regelablauf (RuleProcess)	98
C.1	Umfrage der Statista GCS. Abgerufen am 22.05.2022	X

Tabellenverzeichnis

2.1	Interaktionsparadigmen nach [MINERVA, BIRU und ROTONDI 2015]	13
2.2	Historische Entwicklung vom Internet der Dinge [DURGA, PROF und KUMAR 2020] .	15
2.3	Teilsysteme des Smart Home [STRESE u. a. 2010]	17
2.4	Übertragungsmethoden des Smart Home	24
2.5	Stärken und Schwächen der Home Assistant Plattform	32
3.1	Suchprotokoll des Systematischen Literaturreviews	44
3.2	Einschluss- und Ausschlusskriterien des systematischen Literaturreviews	45
4.1	Funktionale Anforderungen (FAs)	67
4.2	Nicht funktionale Anforderungen (NFAs)	68
4.3	Zusätzliche Anforderungen	69
A.1	Vergleich von MQTT und AMQP [NAIK 2017]	VI
B.1	Vergleich der Plattformen [BRICE 2022] [SUTTNER 2022] [BARCLAY 2020]	VIII
B.2	Vergleich der Plattformen [BRICE 2022] [SUTTNER 2022] [BARCLAY 2020]	IX
E.1	Aufgabenbeschreibung UC 1 - Begrüßung / Check-in	XV
E.2	Aufgabenbeschreibung UC 2 - Notfall-Evakuierung	XVIII

Liste der Code-Beispiele

2.1	Erzeugung und Veröffentlichung einer Nachricht	26
2.2	Empfang und Konsum einer Nachricht	26
2.3	Konstrukt zur Regeldefinition über Home Assistant	33
2.4	Konstrukt zur Regeldefinition über Xbase	40
6.1	Zustandsobjekt als Typ-Variablen	87
6.2	Transformation über eine Switch-Case-Anweisung	90
6.3	Transformation über ein eigenes Objekt	92
F.1	Regeldefinition der Lichtschaltung	XXI
F.2	Regeldefinition der Lichtschaltung über Home Assistant	XXII
F.3	Regeldefinition der Lichtschaltung via openHAB	XXII

Abkürzungsverzeichnis

IoT	Internet of Things	8
IdD	Internet der Dinge	8
SH	Smart Home	3
IT	Informationstechnologie	8
IP	Internetprotokoll	9
IIoT	Industrial Internet of Things, dt. Industrielles Internet der Dinge	12
P2P	Peer to Peer	13
REST	Representational State Transfer	14
SOAP	Simple Object Access Protocol	14
M2M	Machine to Machine	14
API	Application Programming Interface	14
HTML	Hypertext Markup Language	14
XML	Extensible Markup Language	14
JSON	JavaScript Object Notation	14
P&G	Procter & Gamble	15
RFID	Radio-Frequency Identifitaction	15
ITU	International Telecommunication Union	15
IIS	Fraunhofer Institut für Integrierte Schaltungen	15
RE	Requirements Engineering	4
BMWK	Bundesministerium für Wirtschaft und Klimaschutz	21
IFA	Internationale Funkausstellung	22
MQTT	Message Queue Telemetry Transport	24
TCP	Transmission Control Protocol	24
TLS	Transport Layer Security	24
SSL	Secure Sockets Layer	24

AMQP	Advanced Message Queue Protocol	27
HTTP	Hypertext Transfer Protocol	27
CoAP	Constrained Application Protocol	27
PAN	Personal Area Network	28
UI	User Interface	30
DoD	Definition of Done	31
openHAB	open Home Automation Bus	34
OSGi	Open Service Gateway initiatve	34
JVM	Java Virtual Machine	39
JAR	Java Archive	39
SGX	Software Guard Extentions	47
SGCS	Statista Global Consumer Survey	51
AI	Artificial Intelligence	54
KI	Künstliche Intelligenz	54
WLAN	Wireless Local Area Network	54
DSL	Domain Specific Language	40
TORE	Task-oriented Requirements Engineering	59
SOA	Service-Oriented Architecture	85
MVC	Model View Controller	78
MVVM	Model View ViewModel	78
BRMS	Business-Rule-Management-System	49
POJO	Plain Old Java Object	80
CLI	Command Line Interface	106
SPC	Single Point of Contact	69
IDE	Integrated Development Environment	114

Literatur

- ADAM, Sebastian, Norman RIEGEL und Joerg DOERR [2014]. *A Framework for Systematic Requirements Development in Information Systems*. URL: <https://re-magazine.ireb.org/articles/tore> [besucht am 30.05.2022].
- ASCHENDORF, B. [2014]. *Energiemanagement durch Gebäudeautomation: Grundlagen - Technologien - Anwendungen*. Springer Fachmedien Wiesbaden, S. 55–57. ISBN: 9783834820327. URL: <https://books.google.de/books?id=-y8pBAAAQBAJ>.
- BALAKRISHNAN, Sumathi, Hemalata VASUDAVAN und Raja Kumar MURUGESAN [Dez. 2018]. „Smart home technologies: A preliminary review“. In: Association for Computing Machinery, S. 120–127. ISBN: 9781450366298. DOI: 10.1145/3301551.3301575.
- BARCLAY, Lewis [2020]. *Home Assistant vs OpenHAB - Which one is better?* URL: <https://everythingsmarthome.co.uk/home-assistant-vs-openhab-which-one-is-better/> [besucht am 20.05.2022].
- BENDEL, Oliver [2021]. *Smart Home*. URL: <https://wirtschaftslexikon.gabler.de/definition/smart-home-54137> [besucht am 23.03.2022].
- BRAJKOVIC, Sandra [2014]. *Telekom arbeitet am intelligenten Haus der Zukunft*. URL: <https://www.welt.de/wirtschaft/webwelt/article125324076/Telekom-arbeitet-am-intelligenten-Haus-der-Zukunft.html> [besucht am 06.04.2022].
- BRICE, Alex [2022]. *Best of open source smart home: Home Assistant vs OpenHAB*. URL: <https://smarthome.university/your-smart-home-platform-home-assistant-vs-openhab/> [besucht am 01.05.2022].
- BUSCH-JAEGER [2021]. *Busch-Jaeger - Geschichte. 1990er-Jahre - Das intelligente "Haus wird Wirklichkeit*. In: busch-jaeger.de. URL: <https://www.busch-jaeger.de/geschichte/> [besucht am 06.04.2022].
- CAO, Keyan u. a. [2020]. „An Overview on Edge Computing Research“. In: *IEEE Access* 8, S. 85714–85728. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2991734.
- DURGA, P V Vijaya, Asst PROF und T Sujith KUMAR [2020]. „A Literature Survey on Internet of Things“. In: *International Journal for Research in Engineering Application Management (IJREAM) 06*, S. 2454–9150. DOI: 10.35291/2454-9150.2020.0717.

- FRAUNHOFER [2021]. *Historie - Die Geschichte des Fraunhofer-inHaus-Zentrums. Das Fraunhofer-inHaus-Zentrum*. URL: <https://www.inhaus.fraunhofer.de/de/ueber-uns/Historie.html> [besucht am 06.04.2022].
- FUNKE, Holger [2009]. *Das OSGi Framework*. URL: <https://entwickler.de/java/das-osgi-framework> [besucht am 19.06.2022].
- GAMMA, Erich u. a. [1995a]. *Design patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc., S. 325–330. ISBN: 0-201-63361-2.
- [1995b]. *Design patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc., S. 127–134. ISBN: 0-201-63361-2.
- [1995c]. *Design patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc., S. 139–150. ISBN: 0-201-63361-2.
- GARTNER [März 2022]. *Gartner Hype Cycle*. <https://www.gartner.com/en/research/methodologies/gartner-hype-cycle>.
- GILLIS, Alexander [2022]. *What is the internet of things (IoT)*? URL: <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT> [besucht am 25.03.2022].
- HUNKELER, Urs, Hong Linh TRUONG und Andy STANFORD-CLARK [2008]. „MQTT-S - A publish/-subscribe protocol for wireless sensor networks“. In: IEEE Computer Society, S. 791–798. ISBN: 9781424417971. DOI: 10.1109/COMSWA.2008.4554519.
- JOHNSON, Rod [2004]. *Expert one-on-one J2EE design and development*. John Wiley & Sons. ISBN: 0-7645-4385-7.
- KIM, Ji Eun u. a. [2012]. „Seamless integration of heterogeneous devices and access control in smart homes“. In: S. 206–213. ISBN: 9780769547411. DOI: 10.1109/IE.2012.57.
- KITCHENHAM, Barbara und Stuart CHARTERS [Jan. 2007]. „Guidelines for performing Systematic Literature Reviews in Software Engineering“. In: 2.
- KREUZER, Kai [Juni 2014]. *openHAB 1.5 - Release Highlights*. URL: <http://kaikreuzer.blogspot.com> [besucht am 29.04.2022].
- [Jan. 2016]. *openHAB 1.8 and 2.0 beta 1 Release*. URL: <http://kaikreuzer.blogspot.com> [besucht am 30.04.2022].
- [2013]. *openHAB Dokumentation*. URL: <https://github.com/openhab/openhab1-addons/wiki> [besucht am 29.04.2022].
- [2020]. *openHAB Strength*. URL: <https://www.openhab.org/docs/> [besucht am 01.05.2022].
- [Dez. 2012]. *OSGi Connects the World*. URL: <http://kaikreuzer.blogspot.com> [besucht am 29.04.2022].
- [2022]. *OSGi Overview*. URL: <https://www.openhab.org/docs/developer/osgi/osgi.html> [besucht am 01.05.2022].
- KRUCHTEN, Philippe [1995]. *Architectural Blueprints-The "4+1"View Model of Software Architecture*, S. 42–50. DOI: 10.1109/52.469759.

- LASQUETY-REYES, Dr. Jeremiah [2021]. *Statista. Smart Home Report 2021. Statista Digital Market Outlook - Market Report.* (Aritkel-Nr. did-41155-1). URL: <https://de.statista.com/statistik/studie/id/41155/dokument/smart-home-report/> [besucht am 23.05.2022].
- LAZAR, Jonathan, Jinjuan Heidi FENG und Harry HOCHHEISER [2017]. „Chapter 10 - Usability testing“. In: *Research Methods in Human Computer Interaction (Second Edition)*. Hrsg. von Jonathan LAZAR, Jinjuan Heidi FENG und Harry HOCHHEISER. Second Edition. Boston: Morgan Kaufmann, S. 263–298. ISBN: 978-0-12-805390-4. DOI: <https://doi.org/10.1016/B978-0-12-805390-4.00010-8>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128053904000108>.
- LI, Yang u. a. [2018]. „Home Assistant-Based Collaborative Framework of Multi-Sensor Fusion for Social Robot*“. In: ISBN: 9781538673461.
- LIKERT, Rensis [1932]. „A technique for the measurement of attitudes.“ In: *Archives of psychology*.
- LISKOV, Barbara H und Jeannette M WING [2001]. „Behavioral subtyping using invariants and constraints“. In: *Formal methods for distributed processing: a survey of object-oriented approaches*, S. 254–280.
- LUBER, Stefan und Nico LITZEL [2019]. *Definition - Was ist Smart Home.* URL: <https://www.bigdata-insider.de/was-ist-smart-home-a-809018/> [besucht am 08.05.2022].
- [2016]. *Was ist das Internet of Things?* URL: <https://www.bigdata-insider.de/was-ist-das-internet-of-things-a-590806/> [besucht am 22.03.2022].
- MADAKAM, Somayya, R. RAMASWAMY und Siddharth TRIPATHI [2015]. „Internet of Things (IoT): A Literature Review“. In: *Journal of Computer and Communications* 03 [05], S. 164–173. ISSN: 2327-5219. DOI: [10.4236/jcc.2015.35021](https://doi.org/10.4236/jcc.2015.35021).
- MARTIN, Robert C. [2009]. *Clean Code - Refactoring, Patterns, Testen und Techniken für auberen Code*. Hrsg. von Reinhard ENGEL. mitp Verlags GmbH, S. 176–185. ISBN: 978-3-8266-5548-7.
- MARTIN, Robert C [1996]. „The interface segregation principle: One of the many principles of OOD“. In: *C PLUS PLUS REPORT* 8, S. 30–36.
- MARTIN, Robert Cecil [2003]. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, S. 127–131. ISBN: 978-0-13-597444-5.
- MELL, Peter und Timothy GRANCE [Sep. 2011]. *The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology*, S. 2–3. URL: <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>.
- MEYER, Bertrand [1988]. *Object Oriented Software Construction*. Prentice Hall, S. 57–61. ISBN: 978-0-13-629155-8.
- MINERVA, Roberto, Abyi BIRU und Domenico ROTONDI [2015]. „Towards a Definition of the Internet of Things“. In: *Towards a definition of the Internet of Things (IoT)*, S. 59–70. URL: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf.

- MONTANARO, Teodoro u. a. [2021]. „Fast-prototyping approach to design and validate architectures for smart home“. In: *Journal of Communications Software and Systems* 17 [2], S. 177–184. ISSN: 18466079. DOI: 10.24138/jcomss-2021-0005.
- NAIK, Nitin [Okt. 2017]. „Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP“. In: Hrsg. von Nitin NAIK. 2017 IEEE International Systems Engineering Symposium (ISSE), S. 1–6. ISBN: 9781538634035. DOI: 10.1109/SysEng.2017.8088251.
- POHL, K. und C. RUPP [2021]. *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. dpunkt.verlag. ISBN: 9783969102473. URL: <https://books.google.de/books?id=iEgqEAAAQBAJ>.
- PRANZ, Bernhard und Markus SCHILLER [Juni 2018]. „Smart Home with openHAB“. In: S. 2–4.
- ROBSON, Colin [2002]. *Real world research: A resource for social scientists and practitioner-researchers*. Wiley-Blackwell. ISBN: 978-0631213055.
- SCHOUTSEN, Paulus [2020]. *Home Assistant Supervisor*. URL: <https://developers.home-assistant.io/docs/supervisor> [besucht am 01.05.2022].
- SIEPERMANN, Markus und Richard LACKES [2018]. *Internet der Dinge*. URL: <https://wirtschaftslexikon.gabler.de/definition/internet-der-dinge-53187> [besucht am 23.03.2022].
- STRESE, Hartmut u. a. [2010]. *Smart Home in Deutschland. Untersuchung im Rahmen der wissenschaftlichen Begleitung zum Programm Next Generation Media (NGM) des Bundesministeriums für Wirtschaft und Technologie*. Institut für Innovation und Technik (iit). ISBN: 9783897501652.
- SUTTNER, Matthias [2022]. *VERGLEICH: OPENHAB VS. HOME ASSISTANT*. URL: <https://smarthome.msuttner.de/openhab-2/vergleich-openhab-vs-home-assistant/> [besucht am 02.05.2022].
- V., VDE e. [2013]. *Die deutsche Normungs-Roadmap Smart Home + Building*, S. 16–45. URL: www.dke.de.
- VIANI, Federico u. a. [2013]. „Wireless architectures for heterogeneous sensing in smart home applications: Concepts and real implementation“. In: *Proceedings of the IEEE* 101 [11], S. 2381–2396. ISSN: 00189219. DOI: 10.1109/JPROC.2013.2266858.
- WAGNER, Dr. Gunher u. a. [2018]. *Deloitte. Smart Home Consumer Survey 2018. Ausgewählte Ergebnisse für den deutschen Markt*. URL: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/technology-media-telecommunications/Deloitte_TMT_Smart_Home_Studie_18.pdf [besucht am 22.05.2022].
- WU, Chao Lin und Li Chen FU [2012]. „Design and Realization of a Framework for Human–System Interaction in Smart Homes“. In: *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* 42 [1], S. 15–31. ISSN: 15582426. DOI: 10.1109/TSMCA.2011.2159584.
- WUETHERICH, Gerd u. a. [2008]. *Die OSGi Service Platform*. dpunkt.verlag. ISBN: 978-3-89864-457-0.
- [2009]. *Einführung in die OSGi Service Plattform*. URL: <https://nilshartmann.net/uploads/Powerworkshop-OSGi-JAX2009.pdf> [besucht am 19.06.2022].

ZAVALYSHYN, Igor u. a. [Dez. 2020]. „My house, my rules: A private-by-design smart home platform“. In: Association for Computing Machinery, S. 273–282. ISBN: 9781450388405. DOI: 10.1145/3448891.3450333.

Anhang A

Kriterium	MQTT	AMQP
Jahr	1999	2003
Architektur	Client/Broker,	Client/Broker Client/Server
Abstraktion	Publish/Subscribe	Publish/Subscribe Request/response
Header Größe	2 Byte,	8 Byte
Nachrichtengröße	max. 265 MB	Abhängig von Broker/Server
Semantik / Methoden	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close
Zwischenspeicher (Cache) und Proxy	Teilweise	Ja
Standards	OASIS, Eclipse Foundation	OASIS, ISO/IEC
Transport-Protokoll	TCP,	TCP, SCTP
Sicherheit	TLS/SSL	TLS/SSL
Standard-Port	1883/8883 (TLS/SSL)	5671 (TLS/SSL), 5672
Format	binär	Binär
Lizenzmodell	Open Source	Open Source
Support	IBM, Facebook, Eurotech, Cisco, Red Hat Amazon (AWS) etc.	Microsoft, JP Morgan, Bank of America, Barclays, Goldman Sachs Credit Suisse

Tabelle A.1: Vergleich von MQTT und AMQP [NAIK 2017]

Anhang B

Attribut	openHAB	Home Assistant
Architektur	Robustheit und Starrheit, sowie die gewissenhafte Entwicklung	Erfordert häufigere Updates, bietet jedoch eine schnelle Entwicklung und eine viel modernere und anspruchsvollere Architektur [BRICE 2022].
Installation und Wartung	Installation ist genauestens dokumentiert. Initiale Konfiguration muss über die Kommandozeile durchgeführt werden, Updates ebenso.	Ein-Klick-Installationsprozess. Initiale Konfiguration benötigt das Verständnis von YAML-Dateien. Updates können über die UI verwaltet werden [BRICE 2022].
Unterstützte Geräte und Verknüpfungen	Durch die gängigen Smart Home Protokolle werden über 1000 Komponenten unterstützt.	Home Assistant unterstützt ebenso über 1000 Komponenten, wobei die Benutzerfreundlichkeit beim Anlegen und Verwalten bei Home Assistant deutlich angenehmer und leichter erscheint [SUTTNER 2022].
Automatisierungsregeln	Bereitstellung von Automationen durch Xtend. Xtend ist ein flexibler und ausdrucksstarker Java-Dialekt, der sich in Quellcode basierend auf Java 8 kompilieren lässt. Ebenso können über Blockly Automatisierungsregeln erstellt werden. Blockly ist eine clientseitige JavaScript Bibliothek, die visuelle Blöcke mit Anweisungen, Bedingungen uvm. erstellen und editieren lässt. Eine weitere Alternative ist die Verwendung von Node-RED.	Home Assistant bietet die Möglichkeit, Automatisierungen per YAML zu erstellen. Alternativ über ein Node-RED Add-On. YAML-Dateien sind flexibler, jedoch nicht unbedingt benutzerfreundlich.

Tabelle B.1: Vergleich der Plattformen [BRICE 2022] [SUTTNER 2022] [BARCLAY 2020]

Attribut	openHAB	Home Assistant
User Interface	openHAB hat in den User Interface viele Duplikate und zu viele ähnliche Optionen zur Verfügung bei unterschiedlicher Oberfläche.	Die Benutzeroberfläche der Home Assistant Plattform ist für Einsteiger und Anfänger deutlich benutzerfreundlicher und weniger komplex als openHAB.
Mobile Anwendungen	openHAB bietet eine dedizierte App für iOS und Android. Diese ist mit innovativen Optionen ausgestattet.	Home Assistant bietet ebenso eine App für iOS und Android, jedoch bei weitem nicht so stark entwickelt als die von openHAB. Die Benachrichtigungsdienste sind jedoch mit Home Assistant besser genutzt.
Community und Dokumentation	openHAB bietet eine sehr starke und repräsentative Community. Die Dokumentation ist ausführlich und beschreibt die Prozesse, Konzept und Architektur ausführlich.	Home Assistant unterscheidet sich in diesen Punkten nicht von openHAB. Eine starke Community und eine solide Dokumentation der Plattform.
Vergleich von Xtend & YAML	Xtend ist eine sehr mächtige Skriptsprache mit vielen verfügbaren komplexen Strukturen und Funktionen. Dennoch gibt es keine klare Dokumentation, keine Unterstützung von Funktionen, eine seltsame Syntax und kein echtes Tooling.	YAML ist ein lesbarer Datenserialisierungsstandard für Programmiersprachen. Es ist sehr mühsam in größerer Struktur zu verwenden und bieten ebenso keine Funktionen.

Tabelle B.2: Vergleich der Plattformen [BRICE 2022] [SUTTNER 2022] [BARCLAY 2020]

Anhang C

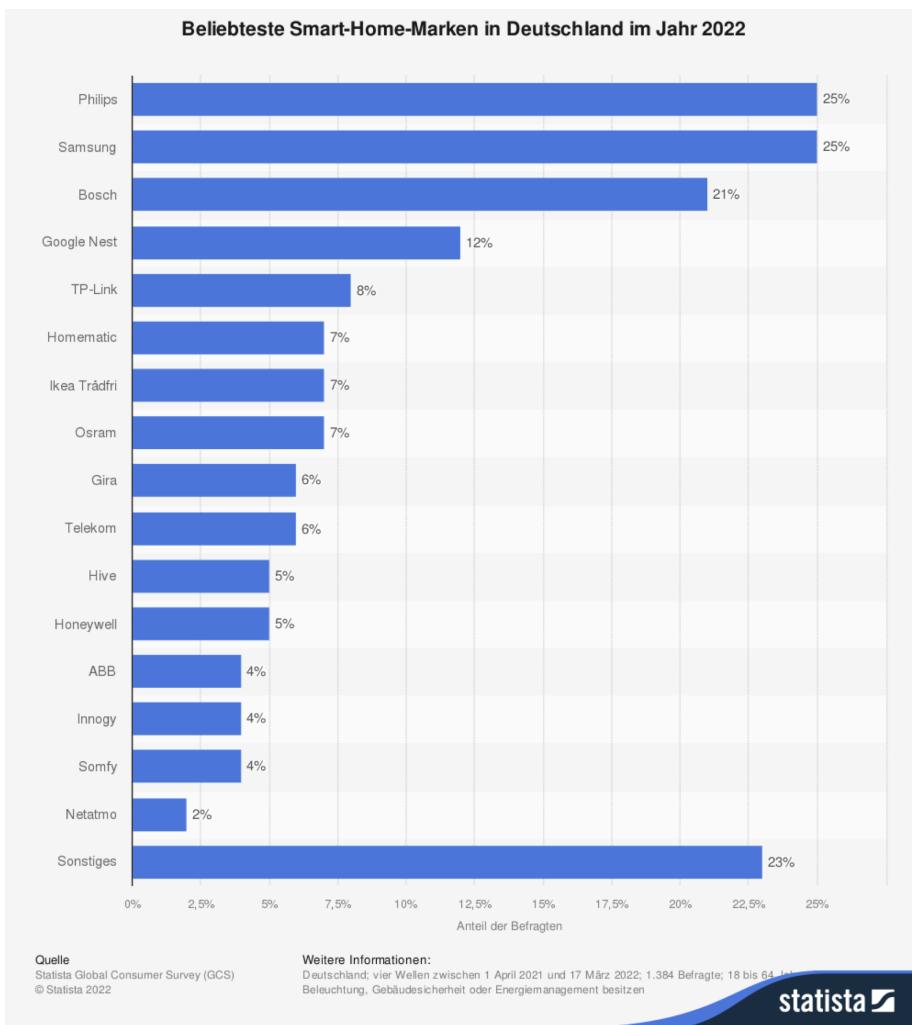


Abbildung C.1: Umfrage der Statista GCS. Abgerufen am 22.05.2022

Anhang D

ROMAN'S PERSONA TEMPLATE



 PICTURE & NAME	 DETAILS	 GOAL
<p>What does the persona look like? What is its name? Choose a realistic and believable picture and name.</p> <div style="text-align: center;">  Maya Zimmermann </div> <p>Picture from unsplash free to use under the Unsplash license</p>	<p>What are the persona's relevant characteristics and behaviours? For instance, demographics, such as age, gender, occupation, and income; psychographics, including lifestyle, social class, and personality; and behavioural attributes like usage patterns, attitudes, and brand loyalty. Only list relevant details.</p> <p>Geschlecht: Weiblich Alter: 28 Beruf: Business and Location Managerin Ausbildung: MBA Business Management Familienstand: Verheiratet, keine Kinder Hobbies: Handball, Ski fahren, Reisen, Malen Attribute: modern, nutzt soziale Medien, innovative Ideen, chaotisch Lebensstil: Typ B (modern, lebt den digitalen Lebensstil) Persönlichkeit: innovativ, emanzipiert, koordiniert Nutzungsmuster: Treibt Innovationen voran, legt Wert auf kontinuierliche und stetige Entwicklung und möglichst geringe Komplexität, nutzt SmartHome Technologien als Anwender</p>	<p>What problem does the persona want to solve or which benefit does the character seek? Why would the persona want to use or buy the product?</p> <ul style="list-style-type: none"> - Sie möchte flexibel in der Erweiterbarkeit der Plattform sein - Neue Funktionen in die Plattform zu integrieren soll wenig komplex sein - Anwendung und Ergänzung von Regeln oder Funktionen sollen schnell und unkompliziert sein - Sie möchte eine innovatives Arbeitsklima erschaffen - Eine zuverlässige Plattform bereitstellen, die Aufgaben übernimmt und Mitarbeiter entlastet. - Sie will selbst mit ihren IT-Kenntnissen Erweiterungen hinzufügen - Einfache Bedienung fördert die Weiterentwicklung und Nutzung der Plattform

www.romanpichler.com
 Template version 02/20

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 Unported license



ROMAN'S PERSONA TEMPLATE



 PICTURE & NAME	 DETAILS	 GOAL
<p>What does the persona look like? What is its name? Choose a realistic and believable picture and name.</p> 	<p>What are the persona's relevant characteristics and behaviours? For instance, demographics, such as age, gender, occupation, and income; psychographics, including lifestyle, social class, and personality; and behavioural attributes like usage patterns, attitudes, and brand loyalty. Only list relevant details.</p> <p>Geschlecht: Männlich Alter: 25 Beruf: Business Analyse Ausbildung: B.Sc. Wirtschaftsinformatik Familienstand: Ledig Hobbies: Games, Schach, Fahrrad fahren Attribute: jung, modern, nutzt soziale Medien, innovative Ideen, chaotisch Lebensstil: Typ A (modern, lebt den digitalen Lebensstil) Persönlichkeit: innovativ, eigenständig, lernfähig und unauffällig Nutzungsmuster: Verwendet gerne die neuesten Technologien der Informatik, Programmiert gerne</p>	<p>What problem does the persona want to solve or which benefit does the character seek? Why would the persona want to use or buy the product?</p> <ul style="list-style-type: none"> - Er möchte Prozesse im Büro automatisieren, sodass er nicht an alles denken muss und sich auf seine Arbeit konzentrieren kann - Er möchte eine innovativeres Arbeitsklima erschaffen - Eine zuverlässige Plattform entwickeln - Er möchte mit der Plattform den Aufwand gering halten, der bei der stetigen Weiterentwicklung entsteht - Er möchte modernste Techniken miteinander vereinen - Ein hat die Vision ein digitalisiertes Gebäude mit aufzubauen

www.romanpichler.com
 Template version 02/20

This work is licensed under a Creative Commons
 Attribution-ShareAlike 4.0 Unported license



ROMAN'S PERSONA TEMPLATE



 PICTURE & NAME	 DETAILS	 GOAL
<p>What does the persona look like? What is its name? Choose a realistic and believable picture and name.</p> <div style="text-align: right; font-size: small; transform: rotate(-90deg);">Picture from unsplash free to use under the Unsplash license</div> <div data-bbox="480 525 871 1107" style="border: 1px solid black; padding: 5px;"> </div> <p>Samantha Nalani Keala</p>	<p>What are the persona's relevant characteristics and behaviours? For instance, demographics, such as age, gender, occupation, and income; psychographics, including lifestyle, social class, and personality; and behavioural attributes like usage patterns, attitudes, and brand loyalty. Only list relevant details.</p> <p>Geschlecht: Weiblich Alter: 42 Beruf: Software Engineer Ausbildung: Diplom Informatiker/in (FH) Familienstand: Verheiratet, ein Kind (männlich im Alter von 7 Jahren) Hobbies: Yoga, Lesen, Reisen Attribute: modern, karriereorientiert, nutzt soziale Medien, innovative Ideen, lernt gerne dazu, ist jeder Aufgabe gewachsen, gerne unter Leuten, strukturiert Lebensstil: Typ A (kulturell interessiert, modern, lebt den digitalen Lebensstil), gestaltet den Alltag und Lebensraum nach ihren Vorstellungen Persönlichkeit: emanzipiert, eigenständig, zielstrebig, erforderlich, neugierig und innovativ Nutzungsmuster: Exklusiv, legt Wert auf Qualität, Performance und Digitalisierung</p>	<p>What problem does the persona want to solve or which benefit does the character seek? Why would the persona want to use or buy the product?</p> <ul style="list-style-type: none"> - Sie möchte eine Struktur in Prozesse bringen, die wiederholbar sind - Das Wohlbefinden im Büro erhöhen - Büroaktivitäten erleichtern - Strom sparen (durch automatischen Ein- und Ausschalten von Lütern, Heizung etc.) - Die Sicherheit im Bürogebäude erhöhen - Alle SmartHome Prozesse zentralisieren und möglichst schnell neue Funktionen und Use Cases abdecken - Mit moderner Technik in Berührung kommen - Möglichst viele Prozesse automatisieren

Anhang E

Use Case 1 - Check-in mit einem Service-Roboter

Aufgabenbeschreibung UC 1

Task:	Begrüßung / Check-In von Member und Gästen
Ziel:	Einen Member (Mitarbeiter) oder einen Guest in der Location mit einem Service-Robot zu begrüßen, wenn dieser eintreten möchte. Nach der Begrüßung findet der Check-In statt, bei dem der Member seinem Platz zugewiesen und der Guest zu seinem Ansprechpartner geleitet wird.
Eingriffs-möglichkeiten:	Auswahl der Situation, Interaktion mit dem Service-Robot (Informieren, Begleiten).
Ursachen:	Terminbuchung, Anmeldung, Klingeln an der Tür, Authentifizierung / Authentisierung an der Tür mittels Biometrische Eingangskontrolle.
Priorität:	Hoch
Durchführungsprofil:	Nach Anmeldung oder Eintritt eines Guests oder Members, Unterbrechungen möglich (Verbindungsfehler, Fehlerhafte Abfrage der Verfügbarkeiten von Platzreservierungen), mittlere Komplexität.
Vorbedingung:	Platzbuchung / Reservierung und Einbuchung für ein Meeting fand im Vorfeld statt, Einverständniserklärung von den Personen liegt vor, Authentisierung über die Eingangskontrolle hat funktioniert / stattgefunden.
Info-In:	Die Person, die eintreten will, ist durch die Buchung oder Authentisierung bekannt, Buchung ist registriert und kann abgerufen werden.
Info-Out:	Begrüßung Standardprotokoll mit Interaktionsmöglichkeiten und Kenntnis des Namens der Person, Check-In Informationen für die einzucheckende Person.
Ressourcen:	Service-Robot, Eintrittskontrolle per Biometrischer Authentisierung, Buchung der Person abrufbar über die API.

Tabelle E.1: Aufgabenbeschreibung UC 1 - Begrüßung / Check-in

UC 1 - User Story

Gast:

Als Guest möchte ich für den Zugang in das Gebäude ein Bild für den Anmeldeprozess hinterlegen können. Vor Ort möchte ich über die Kamera am Eingang mein Gesicht Authentisieren, bei nicht Hinterlegung eines Bildes bei der Anmeldung, möchte ich klingeln. Nach der Authentisierung, bzw. nach Betätigung der Klingel soll sich die Tür öffnen. Beim Hineintreten in das Gebäude nehme ich einen Service-Roboter (Service-Roboter) wahr. Dieser begrüßt mich und startet eine Konversation mit mir. Er bietet mir an die Registrierung, bzw. das Antreffen im Gebäude per QR Code über den Service-Roboter-Bildschirm durchzuführen. Somit werde ich als anwesend markiert und der Mitarbeiter, mit dem ich einen Termin habe, wird informiert. Nach erfolgreicher Registrierung führt mich der Service-Roboter zum Mitarbeiter mit dem ich einen Termin vereinbart habe.

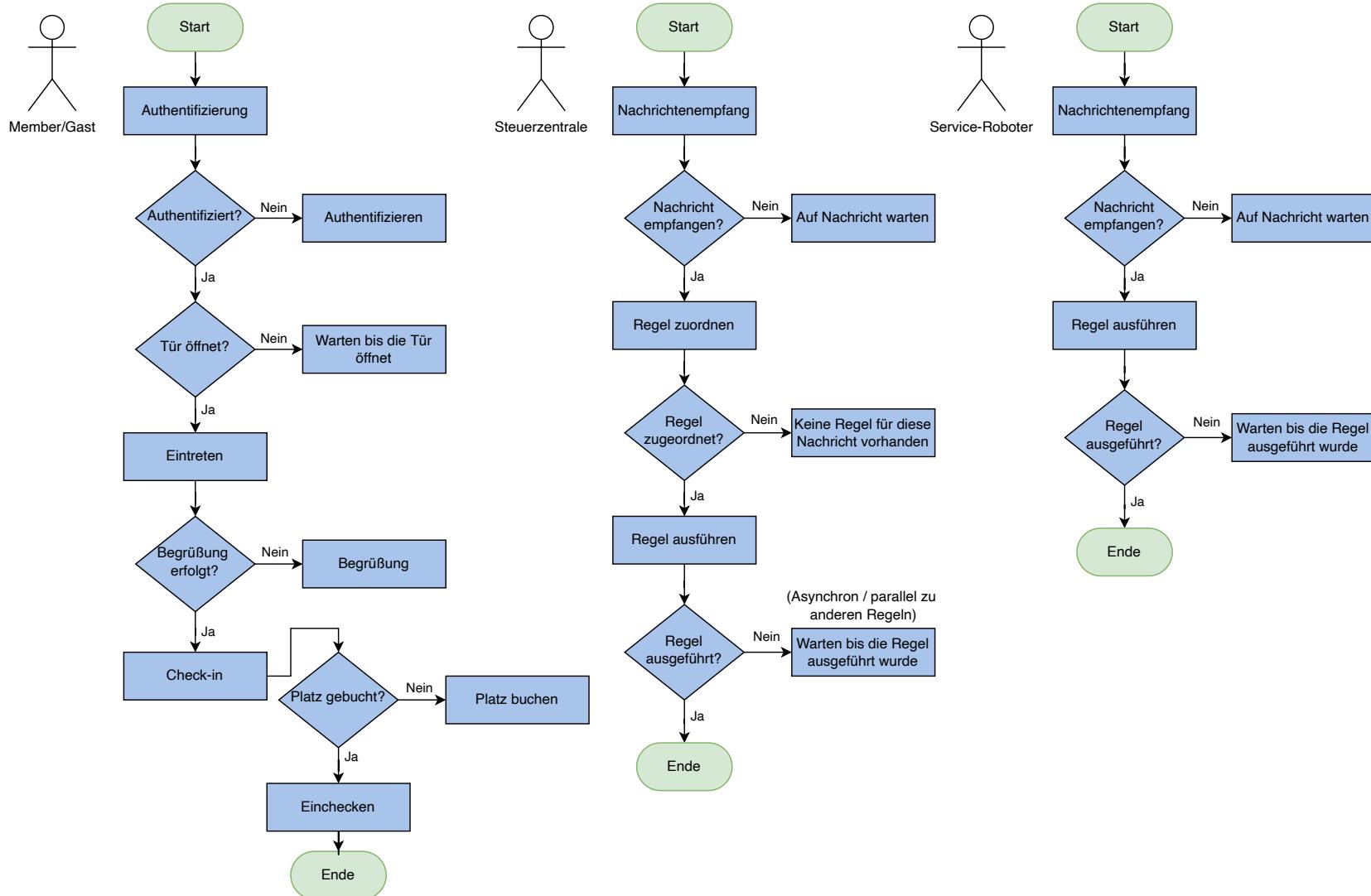
Member:

Als Member möchte ich die Buchung eines Arbeitsplatzes wahrnehmen können. Um in das Gebäude eintreten zu können, ist eine Authentifizierung durch die Gesichtserkennung oder das Abscannen der Chipkarte notwendig. Nach erfolgreicher Autorisierung werde ich von Service-Roboter bei Eintritt des Gebäudes begrüßt. Service-Roboter gibt mir den Arbeitsplatz bekannt und fragt, ob ich zum Platz begleitet werden soll. (oder Service-Roboter mir einen Kaffee an den Arbeitsplatz bringt.) -> (Erweiterung des Use Cases in Planung)

Akzeptanzkriterien

- AK-01: Eine MQTT-MESSAGE mit dem Namen der Authentisierten Person wird empfangen.
- AK-02: Die MQTT-MESSAGE wird gelesen und mittels den Informationen können Abfragen an die DoorJames API erfolgen.
- AK-03: Informationen der DoorJames API werden empfangen und können weiterverarbeitet werden.
- AK-04: Die benötigten Informationen können an den Service-Roboter (Temi) weitergeleitet werden.
- AK-05: Eine MQTT-MESSAGE wird empfangen, sobald der Service-Roboter alle Tasks (ToDo's) des Prozesses abgearbeitet hat.

Use Case - Check-in mit einem Service-Roboter - Ablaufdiagramm



Use Case 2 - Notfall-Evakuierung

Aufgabenbeschreibung UC 2

Task:	Begrüßung / Check-In von Member und Gästen
Ziel:	Einen Notfall zu erkennen und darauf hin alle Member und Gäste, die sich im Büro befinden, informieren und bitten das Gebäude zu verlassen.
Eingriffs-möglichkeiten:	Keine, (Evtl. direkte Interaktion mit dem Service-Roboter).
Ursachen:	Aktivierung des Sensors durch äußerliche Einwirkung.
Priorität:	Hoch
Durchführungsprofil:	nach Meldung eines Notfalls durch Sensoren und Melder
Vorbedingung:	Ein Melder kann Nachrichten veröffentlichen, Topics für die Melder und Empfänger sind definiert.
Info-In:	Meldung eines Sensors (Melder) über einen Notfall (Rauch, Feuer, Wasser, Gas).
Info-Out:	Kenntnis der gebuchten Plätze, Interaktionsmöglichkeiten mit dem Service-Roboter, Benachrichtigung bei Beendung des Prozesses.
Ressourcen:	Service-Roboter, Sensor (Rauch-, Feuer- oder Gas-Melder), Information über aktuelle Buchungen, Zuordnung von Positionen von Büroplätzen.

Tabelle E.2: Aufgabenbeschreibung UC 2 - Notfall-Evakuierung

UC 2 - User Story

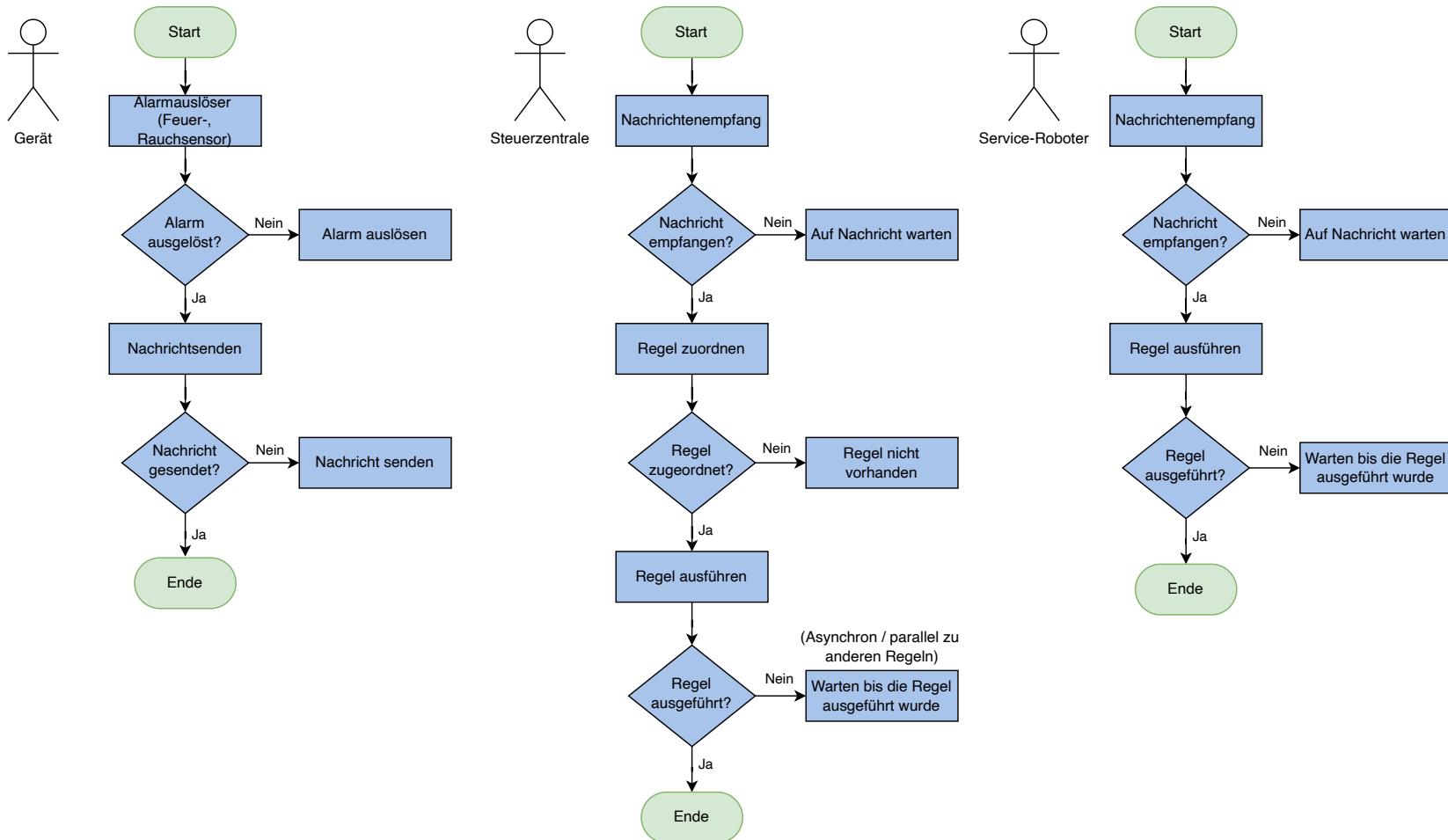
Gast:

Als Gast, Member und Manager möchte ich bei einem Notfall alarmiert werden. Bei eintretendem Fall möchte ich Anweisungen, die zu beachten und umzusetzen sind, erhalten. Ebenso ist ein Hinweis zu den Notausgängen sinnvoll und hilft bei der Flucht, bzw. beim Verlassen des Gebäudes. Damit keine Personen im Büro bleiben, ist eine Information über Anwesende von belangen. Der Member sollte die Möglichkeit haben, den Service-Roboter aufzufordern nach Leuten zu rufen, die nicht in Sichtweite sind. Sobald alle Plätze abgefahrene sind, bzw. durch den Service-Roboter keine Personen mehr erkannt werden, soll eine Benachrichtigung erfolgen, dass alle aus dem Gebäude sind, bzw. keine mehr gesichtet wurden. Diese Information hilft bei der anschließenden Kontrolle über am Treffpunkt anwesender Personen. Abschließend soll ggf. Eine Information über die Platzbuchungen gezeigt werden, damit bei der Kontrolle keine Fehler unterlaufen und keine Personen, die nicht gesehen werden, vergessen werden.

Akzeptanzkriterien

- AK-01: Eine MQTT-Message des Rauchmelders (Sensors) wird von der Steuerzentrale empfangen.
- AK-02: Die MQTT-Message wird gelesen und Abfragen an die DoorJames API erfolgen (Abfrage der Personen zum Zeitpunkt im Bürogebäude)
- AK-03: Informationen der DoorJames API werden empfangen und können weiterverarbeitet werden.
- AK-04: Die benötigten Informationen können an den Service-Roboter (Temi) weitergeleitet werden.
- AK-05: Temi wird losgeschickt, um nach Personen im Gebäude zu suchen.
- AK-06: Die Steuerzenrale schickt eine SMS, E-Mail an die Gebäudeverantwortliche Person, dass ein Notfall losgetreten wurde.
- AK-07: Eine MQTT-Message wird empfangen, sobald der Service-Roboter alle Tasks (ToDo's) des Prozesses abgearbeitet hat.

Use Case - Notfall-Evakuierung mit dem Service-Roboter - Ablaufdiagramm



Anhang F

```
1  @Scope("singleton")
2  public class RuleLampSwitch extends Rule<InnovationLabKA> {
3      ...
4      @Override
5      @TriggerAnnotation
6      public Trigger ruleTrigger(InnovationLabKA state) {
7          return Trigger.MQTT;
8      }
9
10     @Override
11     @ConditionAnnotation
12     public boolean checkCondition(InnovationLabKA state) {
13         return state.isSchalterOn() != state.isLampeOn();
14     }
15
16     @Override
17     @ProcessAnnotation
18     public void ruleProcess(InnovationLabKA state) {
19         logicHubState.setState((state) -> {
20             state.setLampe(state.getSchalter());
21         });
22     }
23     ...
24 }
```

Code-Beispiel F.1: Regeldefinition der Lichtschaltung

```
1   id: '1632733838684'
2   alias: Schalter Kueche Licht An
3   description: ''
4   trigger:
5     - platform: device
6       domain: mqtt
7       device_id: 7281b0009a1b5a48f077dc0873070ce5
8       type: action
9       subtype: 'on'
10      discovery_id: 0x804b50ffe8f57fc action_on
11      condition: []
12    action:
13      - scene: scene.licht_kuche
14    mode: single
```

Code-Beispiel F.2: Regeldefinition der Lichtschaltung über Home Assistant

```
1 rule "Ikea_FB_Button"
2 when Channel "deconz:switch:XXXXXXXX:XXXXXX:buttonevent" triggered
3
4 then
5   if (TRADFRIFernbedienungIKEAofSweden_Button.state==5002){
6     if (IkeaLight.state==ON) {
7       IkeaLight.sendCommand(OFF)
8     }
9     if (IkeaLight.state==OFF) {
10       IkeaLight.sendCommand(ON)
11     }
12   }
13 end
```

Code-Beispiel F.3: Regeldefinition der Lichtschaltung via openHAB

Anhang G

Usability-Test zur Master-Thesis

Konzeption und prototypische Umsetzung einer Steuerzentrale eines smarten Büros mit dem Fokus einer einfachen Handhabung der formalisierten Interaktionen für Softwareentwickler

Im Rahmen der Master-Thesis zu obigem Thema wird ein Usability-Test durchgeführt, anhand dessen die Nutzbarkeit eines Frameworks ermittelt werden soll. Ziel des Systems ist die einfache Handhabung der formalisierte Interaktionen für Softwareentwickler, um in wenigen Schritten eine lauffähige Steuerzentrale und deren Regelwerk zu implementieren.

Die Aufgabe, die innerhalb des Usability-Tests gestellt wird, ist die Implementierung eines kleinen Anwendungsfalls. Dieser wird in kleinere Teilaufgaben unterteilt und Schritt für Schritt erläutert.

Vorab wird das Framework und deren Kernkomponenten erläutert, sodass der Einstieg zur Aufgabe und Nutzung des Frameworks mit einer gewissen Vorkenntnis stattfindet. Nach der Schilderung und der Klärung aller potentiell auftretenden Fragen wird der Test durchgeführt.

Bedingungen:

Es gibt keine zeitliche Einschränkung, die Probanden unterliegen keinem Zeitdruck. Die Durchführung ist ortsunabhängig und wird an unterschiedlicher Hardware durchgeführt. Lediglich ist sicherzustellen, dass die Kommunikation über MQTT mittels einem vorab zur Verfügung gestellten MQTT-Broker erfolgen kann. Hierfür wird ein Raspberry Pi 3 verwendet, der einen vorkonfigurierten MQTT-Broker bereitstellt. Die ein- und ausgehenden MQTT Nachrichten werden, sofern keine ansteuerbaren Geräte zur Verfügung stehen, simuliert. Die Performanz der Kommunikation ist während der Testung zu vernachlässigen. Die im Rahmen dieses Testes benötigten Komponenten, darunter der Service-Roboter sind simuliert und geben entsprechend Rückmeldung.

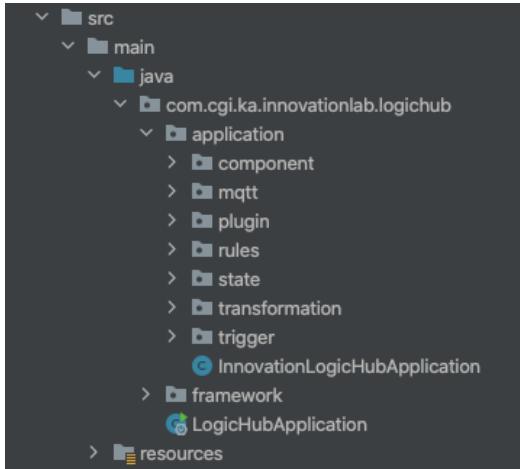
Fragen, die während der Durchführung entstehen, können direkt gestellt werden. Diese werden umgehend beantwortet.

Nach Abschluss des Tests werden die Probanden mittels eines System Usability Scale (SUS) Fragebogen um Rückmeldung gebeten. Abschließend wird noch ein Interview durchgeführt, um weitere Informationen zu erheben. Dabei wird gezielt auf die Erfahrung, sowie die Erweiterung- und ggf. Auf Verbesserungspotentiale eingegangen.



Einleitung zum Usability-Test

Konkrete Kenntnisse über das Spring Framework sind nicht vorauszusetzen. Gestellt ist ein initiales Spring Projekt, welches die Struktur bereits vorgibt.



Die Informationen zur Anbindung des MQTT-Clients an den MQTT-Broker wird begleitend eingerichtet. Unter dem Ordner „resources“ ist in der Konfigurationsdatei die IP, der Nutzernname und das Passwort des clients anzugeben. Weitere Einstellungen müssen nicht getätigt werden. Anschließend sind die Teilaufgaben selbstständig zu lesen und umzusetzen. Diese werden abstrakt gehalten, um Entscheidungen nicht vorweg zu nehmen.

1.1 Vorgegebene Projektstruktur

Die einzige vorgegebene Klasse ist die der Applikation selbst, von dort aus die Interaktionen und das Framework initialisiert und gestartet wird. Es handelt sich dabei um die „InnovationLogicHubApplication“. Diese stellt ein leeres Konstrukt bereit, das es gilt zu füllen, sodass nach Bearbeitung der Aufgaben dieses auch getestet werden kann. Das leere Konstrukt ist der folgenden Abbildung zu entnehmen:

```

1 package com.cgi.ka.innovationlab.logichub.application;
2
3 import org.springframework.stereotype.Service;
4
5 @Service
6 public class InnovationLogicHubApplication {
7
8     public InnovationLogicHubApplication() {
9         System.err.println("Let's tryout the Innovation Lab KA state implementation");
10        System.err.println("-----");
11    }
12}

```

1.2 Konstrukt zu Implementierung des Frameworks

Innerhalb des Konstruktors „InnovationLogicHubApplication()“ sollen die Teilaufgaben implementiert werden. Mit diesen Voraussetzungen kann die Bearbeitung der einzelnen Aufgaben ohne begleitende Hilfe beginnen.



Aufgabe

Es soll ein Anwendungsfall implementiert werden, bei dem ein Service-Roboter einen Mitarbeiter oder Gast, der an der Tür steht, empfangen und begrüßen soll. Eingangsbedingungen sind, dass eine Kamera zur Authentifizierung für die Anwendung simuliert wird. Die MQTT Nachricht wird im Rahmen der Aufgabe manuell erzeugt und losgelöst. Auch die Definition des MQTT Topics ist vorgegeben. Nachdem die simulierte Authentifizierung ausgelöst wurde, soll über die Steuerzentrale eine Regel ausgeführt werden, die folgende Anforderungen erfüllen soll:

- Nach eingehendem Topic soll der Service-Roboter angesteuert und an die Tür geschickt werden. (Die Ansteuerung des Service-Roboters erfolgt ebenso über MQTT. Da in den meisten Fällen kein Roboter verfügbar ist, wird auch diese Kommunikation mittels MQTT simuliert.)
- Ist der Roboter an der Tür, soll er die Begrüßung starten. (Die folgende Interaktion wird im Rahmen des Test nicht durchgeführt. Nach der Nachricht zur simulierten Begrüßung gilt die Aufgabe als erledigt.)

Für die Aufgabe sind folgende Punkte notwendig:

- Die Kenntnis über MQTT Topics, die für die Kommunikation benötigt werden. (Werden im Rahmen des Test vorgegeben)
- Ein Zustandsraum, der alle benötigten Komponenten abbildet.
- Der Service-Roboter als Komponente, sodass dieser bei Ausführung einer Aufgabe für weiteren Aufgaben geblockt werden kann.
- Der Auslöser innerhalb der Regel, welcher durch ein MQTT Trigger (Enum) abgebildet wird.
- Die Transformation der eingehenden MQTT Topics zu Zustandsänderungen, auf die Regeln ausgeführt werden sollen.

Besonders zu beachten ist die Definition von Regeln, sowie deren Integration in das System. Das allgemeine Setup kann weniger gewichtet werden.

Die Vorgehensweise wird den Probanden vor der Durchführung nicht aufgezeigt.

Empfehlung für die Vorgehensweise:

1. Transformation mit den Werten (Topic, Key, Value der JSON, optionale White-List, Name des Attributes im Zustandsraum) definieren.
2. Die Transformation dem Framework übergeben (`logicHub.addTransformation(new SimpleTopicTransformation(...))`)
3. Komponente des Service-Roboters erstellen. („extends Component“)
4. Zustandsraum erstellen, der die Werte als auch Getter und Setter der Werte implementiert. („Z“)
5. Regel definieren, die eine bestimmte Bedingung erfüllen müssen, um darauf eine Aktion auszuführen. („extends Rule<Z>“)
 1. Regel Trigger -> Funktion mit Trigger.MQTT belegen
 2. Regel Condition -> Funktion mit Bedienung des Zustandsraumes überprüfen
 3. Regel Process -> Funktion mit den Schritten zu Begrüßung mittels Thread.sleeps



Fragebogen in Form des SUS

Der folgende Fragebogen soll die Erfahrung und das Empfinden, bzw. die Meinung des Probanden während der Nutzung des Frameworks und der soeben abgeschlossenen Aufgabe widerspiegeln. Hierfür wird das System Usability Scale Template¹ verwendet.

Es sind 10 Aussagen, die mit einem Wert von 0-100 beantwortet werden sollen. 0 stellt dabei die Widersprüche da und 100 die volle Zustimmung.

Nr.	Aussage	Wert
1	Ich denke, dass ich dieses System gerne öfter nutzen würde.	
2	Ich fand das System unnötig komplex.	
3	Ich fand das System einfach zu bedienen.	
4	Ich denke, dass ich die Unterstützung einer technischen Person benötigen würde, um dieses System nutzen zu können.	
5	Ich fand, dass die verschiedenen Funktionen in diesem System gut integriert waren.	
6	Ich dachte, es gäbe zu viele Inkonsistenzen in diesem System.	
7	Ich könnte mir vorstellen, dass die meisten Leute sehr schnell lernen würden, dieses System zu benutzen.	
8	Ich fand das System sehr umständlich zu bedienen.	
9	Ich fühlte mich sehr sicher mit dem System.	
10	Ich musste viele Dinge lernen, bevor ich mit diesem System loslegen konnte.	

* Die Probanden sind anonym und werden zu keinem Zeitpunkt veröffentlicht. Die Bewertung durch den Fragebogen ist ebenso anonym. Es kann zu keinem Zeitpunkt eine Schlussfolgerung auf die Probanden gezogen werden. Die Durchführung des Usability-Tests, sowie die Auswertung des Fragebogens gilt nur zu Erhebung von Informationen im Rahmen der Nutzbarkeit.

¹ <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>



Erklärung

Ich versichere, dass ich diese Master-Thesis mit dem Thema: „*Konzeption und prototypische Umsetzung einer Steuerzentrale eines smarten Büros mit dem Fokus einer einfachen Handhabung der formalisierten Interaktionen für Softwareentwickler*“ selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlichen oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe. Die Arbeit wurde noch keiner Kommission zur Prüfung vorgelegt und verletzt in keiner Weise Rechte Dritter.

Ort, Datum

Unterschrift