



# **Digitale Transformation mit SAP Leonardo in der Energiewirtschaft**

Bachelorarbeit

Erstprüfer: Prof. Dr. Hergen Pargmann  
Zweitprüfer: Prof. Dr. Harald Schallner

Vorgelegt von: Kübra Tokuc  
Scharnhorststraße 54  
26131 Oldeburg  
+49 1577 266 1219  
kuebra.tokuc@student.jade-hs.de

Abgabetermin: 20. Januar 2020

# Inhaltsverzeichnis

<b>Akronyme</b>	<b>IV</b>
<b>Abbildungsverzeichnis</b>	<b>VI</b>
<b>Tabellenverzeichnis</b>	<b>VII</b>
<b>Quelltextverzeichnis</b>	<b>VII</b>
<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Problemstellung . . . . .	2
1.3. Lösungsansatz . . . . .	3
1.4. Aufbau der Arbeit . . . . .	3
<b>2. Grundlagen</b>	<b>5</b>
2.1. Industrie 4.0 . . . . .	5
2.1.1. Definition . . . . .	5
2.1.2. Historischer Kontext . . . . .	6
2.1.3. Technologische Treiber . . . . .	8
2.1.4. Kommunikationssysteme . . . . .	11
2.2. Digitale Transformation mit Internet of Things . . . . .	12
2.2.1. Der Wandel im Energiesektor . . . . .	12
2.2.2. Die Herausforderung IoT . . . . .	15
2.2.3. Vereinheitlichung durch Referenzarchitektur . . . . .	16
2.2.4. Das neue Paradigma: Cloud Computing . . . . .	18
2.3. Werkzeuge für die Transformation . . . . .	20
2.3.1. SAP Cloud Platform . . . . .	21
2.3.2. SAP Leonardo . . . . .	22
2.3.3. Amazon Web Services . . . . .	24
<b>3. Umsetzungskonzept für die digitale Transformation</b>	<b>25</b>
3.1. Repräsentativer Anwendungsfall für die Energiewirtschaft . . . . .	25
3.1.1. Ausgangsszenario . . . . .	25
3.1.2. Anforderungserhebung . . . . .	27
3.2. Anforderungsanalyse . . . . .	28
3.2.1. Kontextebene . . . . .	28
3.2.2. Systemebene . . . . .	32
3.2.3. Technische Ebene . . . . .	36
3.3. Systemanalyse und -entwurf . . . . .	39
3.3.1. Systemarchitektur . . . . .	39

3.3.2. Datentransport: Internet of Things Gateway . . . . .	40
3.3.3. Datenhaltung: SCP Internet of Things Service . . . . .	41
3.3.4. Analyse: SAP Leonardo IoT . . . . .	43
3.3.5. Destinations . . . . .	44
3.3.6. Sicherheit . . . . .	44
3.3.7. Kompatibilität mit Referenzarchitektur . . . . .	45
3.3.8. Systementwurf gemäß Architekturkonzept . . . . .	46
3.4. Prototypische Implementierung des Anwendungsfalls . . . . .	48
3.4.1. Erzeugung eines cyberphysischen Systems . . . . .	48
3.4.2. Einrichtung der Gateway Edge . . . . .	49
3.4.3. Registrierung der Geräte . . . . .	51
3.4.4. Datentransfer an die Cloud . . . . .	52
3.4.5. Erzeugung des digitalen Zwillings . . . . .	53
3.4.6. Visualisierung in einer UI5-Anwendung . . . . .	55
3.4.7. Regeln, Ereignisse und Aktionen . . . . .	58
3.4.8. Zusammenfassung der Implementierung . . . . .	64
<b>4. Evaluation</b>	<b>65</b>
4.1. Evaluation auf Kontextebene . . . . .	65
4.2. Evaluation auf Systemebene . . . . .	66
4.3. Evaluation auf technischer Ebene . . . . .	67
4.4. Handlungsempfehlung . . . . .	68
<b>5. Kritische Würdigung und Ausblick</b>	<b>70</b>
<b>Literatur</b>	<b>V</b>
<b>A. Anforderungen</b>	<b>X</b>
A.1. Anforderungen aus Kontextebene . . . . .	X
A.2. Lösung aus Kontextebene . . . . .	XI
A.3. Anforderungen aus Systemebene . . . . .	XI
<b>B. Quelltext</b>	<b>XIII</b>
B.1. Python Skript: test_data.py . . . . .	XIII
B.2. Python Skript: send_test_data.py . . . . .	XVI
B.3. Python Skript: AWS_Flask.py . . . . .	XVIII
B.4. API-Anfragen für Aktionen . . . . .	XX
<b>C. Notizen, die ich später noch gebrauchen könnte</b>	<b>XXIII</b>

# Akronyme

<b>IKT</b>	Informations- und Kommunikationstechnologien .....	1
<b>IoT</b>	Internet of Things .....	1
<b>REST</b>	Representational State Transfer .....	11
<b>MQTT</b>	Message Queuing Telemetry Transport .....	11
<b>OPC-UA</b>	Open Platform Communications Unified Architecture.....	12
<b>URI</b>	Unified Resource Identifier .....	12
<b>SOA</b>	service-orientierte Architektur .....	12
<b>HTTP</b>	Hypertext Transfer Protocol.....	11
<b>CMS</b>	Condition Monitoring System .....	14
<b>ERP</b>	Enterprise-Resource-Planning .....	23
<b>AWS</b>	Amazon Web Services .....	21
<b>GCP</b>	Google Cloud Platform.....	21
<b>SCP</b>	SAP Cloud Platform .....	21
<b>API</b>	Application Programming Interface.....	23
<b>SNS</b>	Simple Notification Service .....	24
<b>CPS</b>	Cyber-physische Systeme .....	6
<b>IPv4</b>	Internet Protocol Version 4.....	8
<b>IPv6</b>	Internet Protocol Version 6.....	8
<b>RFID</b>	radio-frequency identification .....	8
<b>RAMI 4.0</b>	Referenzarchitekturmodell Industrie 4.0 .....	31
<b>IaaS</b>	Infrastructure as a service .....	18
<b>PaaS</b>	Platform as a service .....	18
<b>SaaS</b>	Software as a service .....	19
<b>AWS</b>	Amazon Web Services .....	21
<b>SCADA</b>	Supervisory Control and Data Acquisition .....	25
<b>EVU</b>	Energieversorgungsunternehmen .....	27
<b>HANA</b>	High Performance Analytic Appliance .....	26
<b>PAL</b>	Problem-Anforderung-Lösung .....	27
<b>I-4.0-K</b>	Industrie-4.0-Komponente .....	27

<b>JSON</b>	JavaScript Object Notation.....	41
<b>Protobuf</b>	Protocol Buffers.....	41
<b>TLS</b>	Transport Layer Security .....	45
<b>CLI</b>	Command Line Interface .....	60
<b>SA</b>	Systemadministrator	

## **Abbildungsverzeichnis**

1.	Die vier Stufen Industrieller Revolutionen . . . . .	7
2.	CPS in der Industrie 4.0 . . . . .	9
3.	Prognose zur Anzahl der vernetzten Geräte im Internet of TThings weltweit . . . . .	10
4.	Dimensionen der Digitalisierung . . . . .	15
5.	IT-sicht und Industrie-4.0-Komponente . . . . .	17
6.	Die Cloud-Service-Modelle . . . . .	20
7.	Systemabgrenzung und Systemkontext . . . . .	28
8.	Use Case Diagramm der Kontextebene . . . . .	31
9.	Datenflussdiagramm . . . . .	34
10.	Erweitertes Use Case Diagramm auf Systemebene . . . . .	35
11.	Architektur von SAP . . . . .	40
12.	Gerätemodell . . . . .	42
13.	Architekturübersicht SAP Leonardo IoT . . . . .	44
14.	Referenz zu den Schichten der RAMI 4.0 . . . . .	45
15.	Systementwurf . . . . .	47
16.	Cyberphysisches System für die Simulation . . . . .	49
17.	Gerätemodell der erstellten Instanz . . . . .	51
18.	Visualisierung der empfangenen Daten in Tabellenform . . . . .	53
19.	Mapping zwischen Thing und Sensor . . . . .	55
20.	Startseite der Anwendung . . . . .	56
21.	Details zur Anlage . . . . .	57
22.	Analyseseite . . . . .	58
23.	Definition einer Regel . . . . .	59
24.	Übersicht über den Zustand der Anlage . . . . .	63
25.	Detaillierte Auflistung der Ereignisse . . . . .	64
26.	Empfang der Benachrichtigungs-SMS . . . . .	67
27.	Das Referenzarchitekturmodell Industrie 4.0 . . . . .	XXIII
28.	Ebenen der Industrie-4.0-Komponente . . . . .	XXV

## **Tabellenverzeichnis**

1.	Das PAL-Modell . . . . .	29
3.	Probleme aus Kontextebene . . . . .	30
5.	Probleme aus Systemebene . . . . .	33
6.	Probleme aus technischer Ebene . . . . .	36
7.	Anforderungen aus technischer Ebene . . . . .	38
8.	Gateway-Protokolle . . . . .	41

10.	Anforderungen aus Kontextebene . . . . .	X
12.	Lösungen aus Kontextebene . . . . .	XI
13.	Anforderungen aus Systemebene . . . . .	XIII

## Listings

1.	API-Endpunkte der Gateways . . . . .	41
2.	Gateway-Verbindung zur Cloud . . . . .	50
3.	Zieladresse für Sensorwerte . . . . .	50
4.	GET-Anfrage für einen Client-Key . . . . .	50
5.	Gateway-Eigenschaften . . . . .	50
6.	Das Data-Feld der POST-Anfrage . . . . .	52
7.	Flask-API für AWS SNS . . . . .	60
8.	Deployment in die SAP Cloud Platform . . . . .	61
9.	JSON Payload an Destination . . . . .	61
10.	POST-Anfrage zum Senden den Befehls . . . . .	61
11.	GET-Anfrage für Commands an das Gateway . . . . .	62
12.	Beispielanfrage für die Generierung eines Ereignisses . . . . .	62

# 1. Einleitung

Internet of Things (IoT)

## 1.1. Motivation

*„Wenn Technologien und Gesellschaft sich schneller ändern, als Unternehmen in der Lage sind sich anzupassen, dann kommt es ganz nach den Regeln der Evolution zum Aussterben bestimmter Unternehmensarten.“*

Roth (2016, S. 3, zitiert nach Land, K.-H. 2015)

Die vierte industrielle Revolution durchläuft zahlreiche Branchen und bringt das Potenzial, sie grundlegend zu verändern. Vielen Branchen und Unternehmenstypen eröffnen sich Chancen, durch die Unterstützung von Informations- und Kommunikationstechnologien (IKT) ihre Produktivität zu steigern. Gleichzeitig wird beobachtet, wie andere etablierte Geschäftsstrukturen von digitalen Geschäftsmodellen vertrieben werden (Lauenroth et al., 2016). Andere Branchen wiederum sind vollständig abhängig von den Technologien der Industrie 4.0. Vor allem die Energiewirtschaft ist im Zuge der Energiewende einem grundlegenden Transformationsprozess unterworfen. Durch den Umstieg von Energieproduktionsmethoden wie Kernkraft oder Kohlekraft zu erneuerbaren Energien verlagert sich die Sicht von zentraler Produktion auf dezentrale Produktion (Doleski, 2016a). Der Paradigmenwechsel in der Produktion geht in diesem Fall parallel mit Entwicklungen in den IKT einher. Ohne die technologischen Treiber der Industrie 4.0 wäre die Koordination der dezentralen Anlagen sowie die Einspeisung der Energie in das Netz nicht möglich (Utecht und Zierau, 2018). Dabei nimmt die Geschwindigkeit, in der neue Technologien entwickelt werden, rasant zu. Die industrielle Welt wird in ihren kleinsten Komponenten immer mehr vernetzt und nimmt auch in Entscheidungs- und Steuerungsprozessen ein immer höheres Tempo an. Physische Objekte migrieren in die virtuelle Welt und bringen das Potenzial, ihren Wert zu steigern, indem sie Daten über ihren Zustand melden. Produktionsanlagen werden zu *cyberphysischen Systemen*, die sich intelligent mit Teilnehmern der gesamten Wertschöpfungskette vernetzen können (Lauenroth et al., 2016). Es entsteht ein *Internet der Dinge und Dienste*.

## **1.2. Problemstellung**

Im Kontext dieser Arbeit bildet die Motivation gleichzeitig das Problem. Um in der digitalisierten Welt als Unternehmen überleben zu können und nicht von der Geschwindigkeit überholt zu werden, muss ein Umdenkungsprozess stattfinden (Lauenroth et al., 2016). Besonders rund um die Energiewirtschaft darf nicht gefragt werden, ob man dem Trend der Digitalisierung folgen sollte. Stattdessen muss gefragt werden, mit welchen Mitteln der Transformationsprozess am besten gelöst werden kann und welche Chancen sich daraus ergeben können (Utecht und Zierau, 2018). Das Prinzip „every budget is an IT-budget“ (Lauenroth et al., 2016, S. 5) beschreibt das Phänomen der Verschmelzung von Mechanik und Informationstechnik, von Maschinenbau und Software. Vor allem aufgrund der Veränderungen der Rahmenbedingungen in der Energiewirtschaft ist dieser Gedanke weiterzuführen. Auch wenn bereits Technologien zur automatisierten Steuerung und Überwachung der dezentralen Anlagen zur erneuerbaren Energieproduktion existieren, bilden diese oft nur Insellösungen. Die Möglichkeiten zur Erschließung neuer Geschäftsfelder durch die Erstellung von IT-Dienstleistungen, oder zur Gewährleistung der Produktqualität benötigen eine integrierte zentrale Datenplattform für den gesamten Wertschöpfungsprozess (Utecht und Zierau, 2018). Vor allem durch die Nutzung von cloudbasierten Innovationsplattformen können Chancen entstehen, in Echtzeit auf Probleme zu reagieren und intelligente Dienste für Kunden und Geschäftspartner bereitzustellen. Als Innovationsplattform soll im Rahmen dieser Arbeit die Eignung von SAP Leonardo geprüft werden. Aus diesem Grund ergeben sich für diese Arbeit folgende Forschungsfragen:

### **FF1 Wie kann SAP Leonardo die digitale Transformation in der Energiewirtschaft mit Internet of Things unterstützen?**

- FF1.1 Welche Anforderungen an ein System ergeben sich aus Sicht der dezentralen Ernergieerzeugung?
- FF1.2 Welche Möglichkeiten zur intelligenten Vernetzung bietet die zugrundeliegende Systemarchitektur?
- FF1.3 Mit welcher Systemarchitektur können die Anforderungen aus FF1.1 erfüllt werden?

### **1.3. Lösungsansatz**

Für die Beantwortung der Forschungsfragen wird ein Prototyp - hardwareseitig mit einem Raspberry Pi und softwareseitig mit SAP Leonardo IoT - entwickelt und evaluiert. Es wird ein cyberphysisches System als Simulation für eine Produktionsanlage erzeugt, welches durch den *digitalen Zwilling* in ein *IoT-Netzwerk* integriert wird. In dem Netzwerk soll das CPS mit weiteren Teilnehmern des Systems kommunizieren und interagieren. Die Kommunikation beläuft sich unter anderem auf das Senden von Zustandsdaten, welche analysiert und visuell bereitgestellt werden. Ziel des Prototypen ist, neben der Behandlung der Forschungsfragen, aufzuzeigen, wie man durch die Innovationsplattform SAP Leonardo die Produktqualität gewährleisten und IT-Dienstleistung erzeugen kann. Als Basis für die Prototypentwicklung wird ein repräsentativer Anwendungsfall aus Sicht der Energiewirtschaft entwickelt. Für den Anwendungsfall werden nach dem PAL-Modell (nach Lauenroth et al. (2016)) auf verschiedenen Abstraktionsebenen Problemräume definiert, Anforderungen erhoben und Lösungen entwickelt. Die Anforderungserhebung für das Gesamtsystem basiert auf der Eingrenzung durch die Beantwortung der Frage FF1.1. Da die Vernetzung in *IoT-Netzwerken* oft über verschiedene Wertschöpfungsstufen und Unternehmenstypen hinweg stattfindet, wurde in der Anforderungserhebung die Referenzarchitektur der Bundesregierung, Wirtschaft und Wissenschaft als Anforderungsquelle einbezogen. Weil sich eine prototypische Entwicklung aber nur auf wesentliche Konzepte beschränkt, wird die Forschungsfrage FF1.2 in einer ausführlichen Analyse der SAP Leonardo Systemlandschaft behandelt. Schließlich wird mit dem Systementwurf und der Implementierung der Forschungsfrage FF1.3 begegnet.

### **1.4. Aufbau der Arbeit**

- Zunächst Industrie 4.0 und treibende Faktoren allgemein
- Was ist der Mehrwert von Kommunikationssystemen und welche Protokolle sind Grundlage für die Vernetzung?
- Welche Referenzarchitektur vereinheitlicht industrielle Standards und Anforderungen an die Systeme?

- Was ist Cloud Computing und welche Rolle spielen dessen Technologien für Industrie 4.0?
- Welche Toolsets sind für die Lösung vorhanden?
- Use Case: Für welchen Anwendungsfall in der Energiebranche wird ein Prototyp entwickelt?
- Was sind die Anforderungen an den Prototypen? Näherer Bezug auf Energiebranche.
- Welche Komponenten besitzt das entworfene System bzw. sind notwendig?
- Wie sieht die Implementierung im Detail aus?
- Evaluierung des Vorgehens
- Fazit

## 2. Grundlagen

### 2.1. Industrie 4.0

Dieses Kapitel soll die Relevanz der Thematik verdeutlichen, indem sie in einen gesellschaftlichen und wirtschaftlichen sowie in einen technischen Kontext gebracht wird. Zunächst wird erläutert, was hinter dem Begriff *Industrie 4.0* steckt. Dem Ausdruck wird mehr Sinn verliehen, wenn die historische Entwicklung bekannt ist. Anschließend werden die treibenden Technologien kurz erläutert. Da die Vernetzung für Industrie 4.0 eine tragende Rolle spielt, werden zuletzt Theorien und Technologien zu Kommunikationssystemen aufgegriffen.

#### 2.1.1. Definition

Laut der Fraunhofer-Gesellschaft (2016) habe die *Industrie 4.0* einen revolutionären Einfluss auf die Wertschöpfung in der Industrie und somit auf die Volkswirtschaft. Dieser Marketingbegriff prägt heute die Agenda vieler Unternehmen und Forschungseinrichtungen. Doch was genau hinter dem Begriff zu verstehen ist, bleibt aufgrund des Fehlens einer „wissenschaftlichen Präzision“ uneindeutig (Bendel, 2019b). Für die Gestaltung der digitalen Transformation entstand das Netzwerk *Plattform Industrie 4.0* zwischen der Bundesregierung, Forschungseinrichtungen und Wirtschaft. Dieses hat zum Ziel, die Produktion mittels modernster Informations- und Kommunikationstechnologien entlang der Wertschöpfungskette „flexibler, individueller und effizienter“ gestalten (Bundesministerium für Wirtschaft und Energie, 2019). In der Umsetzungstrategie der Bitkom e.V (2015, S. 8) wird der Begriff wie folgt definiert:

„Der Begriff *Industrie 4.0* steht für die vierte industrielle Revolution, einer neuen Stufe der Organisation und Steuerung der gesamten Wertschöpfungskette über den Lebenszyklus von Produkten. Dieser Zyklus orientiert an den zunehmend individualisierten Kundenwünschen und erstreckt sich von der Idee, dem Auftrag über die Entwicklung und Fertigung, die Auslieferung eines Produkts an den Endkunden bis hin zum Recycling einschließlich der damit verbundenen Dienstleistungen.“

Darüber, auf welcher Basis die digitale Transformation in der Industrie stattfinden wird, scheinen sich jedoch alle einig: durch die *intelligente Vernetzung aller*

*am Produktlebenszyklus beteiligten Menschen, Objekte und Systeme* (Roth, 2016). Das Wesentliche der Vernetzung bilden dezentrale Cyber-physische Systeme (CPS) (Bendel, 2019a). Die tatsächliche Wertschöpfung ergibt sich aus den in Echtzeit verfügbaren quantitativen Informationen, aus welchen man durch Analysen qualitative Erkenntnisse schließen und optimierte Aktionen auslösen kann (Hänisch, 2017). Nach Sendler (2016) gibt es für den Erfolg von *Industrie 4.0* entscheidende Faktoren: Mittlerweile sind digitale Komponenten wie Sensoren, Aktoren oder Kameras so günstig und klein, dass sie in allen möglichen Bereichen eingebaut werden und Umgebungsdaten messen und aufnehmen können. Dank dem Internetprotokoll IPv6 und dem dadurch verfügbaren Adressraum können diese Komponenten ihren Platz im Internet finden. Dass die Informatik sich damit zur wichtigsten Ingenieursdisziplin entwickle, sei unterlässlich, da sie für die Vernetzung der Welt gebraucht werde.

### **2.1.2. Historischer Kontext**

Um den aktuellen Stellenwert von Industrie 4.0 zu beschreiben, wird oft von der vierten industriellen *Revolution* gesprochen. Revolutioniert wurde die Industrie erstmalig im 18. Jahrhundert mit der Erfindung der Dampfmaschine durch Thomas Newcomen und James Watt - die **erste industrielle Revolution** (Roth, 2016). Mit Errungenschaften wie dem dampfgetriebenen Webstuhl ging eine **Mechanisierung** der Produktion einher. Schon damals förderte eine Erfindung, die Lokomotive, eine Vernetzung, die einen regen Warenaustausch ermöglichte (Barthelmäss et al., 2017).

Durch die **Elektrifizierung** in der Industrie und der Zerlegung von Produktions schritten in einzelne Einheiten konnten ab 1870 die Waren auf Fließ- und Förderbändern in Massen produziert werden. Angestoßen wurde die **zweite industrielle Revolution** von Erfindungen wie der Verbrennungskraftmaschine und dem Elektromotor sowie der Herstellung von Syntheseprodukten. Neben fossilen Energieträgern wie Kohle und Öl kam auch die Kernkraft hinzu (Barthelmäss et al., 2017).

Die **dritte industrielle Revolution** ab den 1970er Jahren, in der wir uns noch heute befinden, brachte die **Automatisierung** der Produktion durch die **Digitalisierung** (Voigt, 2018). Getrieben wurde die Revolution durch das Wirtschaftswunder der 1960er Jahre (Roth, 2016) und ermöglicht durch den Ausbau von Informations- und Kommunikationstechnologien. Entscheidende Technologien waren vielfältig. 1941

entwickelte der Bauingenieur Konrad Zuse den ersten programmgesteuerten und vollautomatischen Computer und setzte den Grundstein für eine rasante Entwicklung der nachfolgenden Technologien. Mit der Vebreitung von Mikroprozessoren, der Miniatisierung der Elektronik sowie der nach dem Mooreschen Gesetz vorausgesagten Zunahme der Prozessorstärke nahm die Welt ein neues Tempo an (Sendler, 2016). Einen nicht unwesentlichen Beitrag leistet die Raumfahrttechnik, ohne deren Satellitentechnik eine globale Kommunikation nicht möglich wäre. Da der energieintensive Einsatz dieser Technologien ein Bewusstsein über die Endlichkeit der fossilen Ressourcen schuf, kamen auch erneuerbare Energien hinzu (Barthelmä et al., 2017).

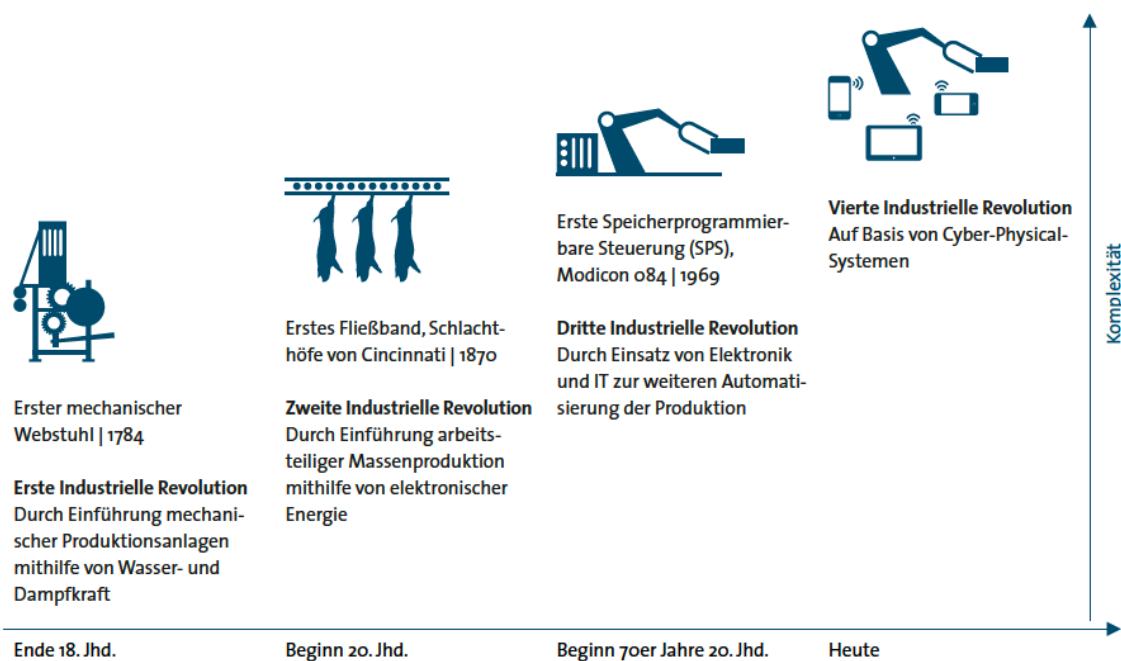


Abbildung 1: Die vier Stufen Industrieller Revolutionen (Bauer et al., 2014, S. 9)

Die Industrie 4.0 tatsächlich als **vierte Industrielle Revolution** zu bezeichnen wird kritisiert, da sie u.a. keine neuen technologischen Innovationen hervorbringe, sondern sich lediglich an den Technologien der dritten Revolution bediene (Barthelmä et al., 2017). Auch wenn die Technologien nicht unbedingt als revolutionär zu bezeichnen sind, durchlaufen sie eine Evolution und stoßen einen Wandel durch Industrie 4.0 an (Roth, 2016). Es entstehen eine Vielzahl neuer Geschäftsmodelle und Produktionsprozesse, die zu Effizienz- und Produktivitätssteigerungen führen

(Bitkom e.V, 2015).

### 2.1.3. Technologische Treiber

Industrie 4.0 zeichnet sich durch das Zusammenwachsen der realen und physischen Welt zu mit Sensorik und Aktorik ausgestatten Objekten aus, die per Internet miteinander verbunden sind (Bitkom e.V, 2015). Dieses Phänomen ist geprägt von Trendtechnologien wie *Big Data*, *Internet of Things*, *Blockchain*, *Cloud Computing* und *Machine Learning*, die in kombinierter Nutzung einen Mehrwert erzeugen.

**Cyber-physische Systeme (CPS)** sind physische Objekte mit einem Datenobekt als virtuelle Präsenz im Netz und bilden die Grundlage von Industrie 4.0 (Drath, 2016). Das Hauptmerkmal eines CPS ist das mit dem Internet verbundene *eingebettete System*, welches über Sensoren Daten aus der Umwelt aufnimmt, über Aktoren wieder mit ihr interagiert und sich somit an sie anpasst. Diese Fähigkeit, die Informationen zu verarbeiten und zu versenden, wird dem CPS durch *Ubiquitous Computing* verliehen. Entscheidend für die implizite und allgegenwärtige Nutzung von IT ist neben der Austattung mit Sensoren und Aktoren die Verfügbarkeit von Kommunikationsmodulen und Rechenleistung (Roth, 2016). Somit wird den Systemen die Fähigkeit verliehen, sich untereinander *dezentral und autonom* zu vernetzen (Bauernhansl, 2014). Zudem besitzen CPS die Eigenschaft, auch mit Menschen zu interagieren: zum einen über direkte Kommunikation wie Monitoring und Befehle und zum anderen, um komplexe Aufgaben gemeinsam zu lösen (Lüth, 2016).

**Das Internet of Things (IoT)** ist das Verbindungsstück zwischen dem Internet und dem Objekt des *Ubiquitous Computing* (Roth, 2016). Das Gerät bzw. das *Thing* ist in der Lage dazu, selbstständig den eigenen Zustand zu erfassen und zu kommunizieren (Kenn, 2016). Dafür muss das Gerät allerdings eindeutig mit z.B. IP-Adressen oder radio-frequency identification (RFID) identifizierbar sein. Während das Internet Protocol Version 4 (IPv4) mit seinen rund 4,3 Milliarden Adressen noch nicht einmal die in 2016 verbundenen 6 Milliarden Geräte (siehe Abbildung 3) platzieren kann, schafft das Internet Protocol Version 6 (IPv6) mit 340 Sextillionen (2 hoch 128) Adressen genügend Platz für 20 Milliarden prognostizierte Geräte in 2020. Das Entscheidende für das *Internet* der Dinge ist, dass die Daten nicht lokal auf

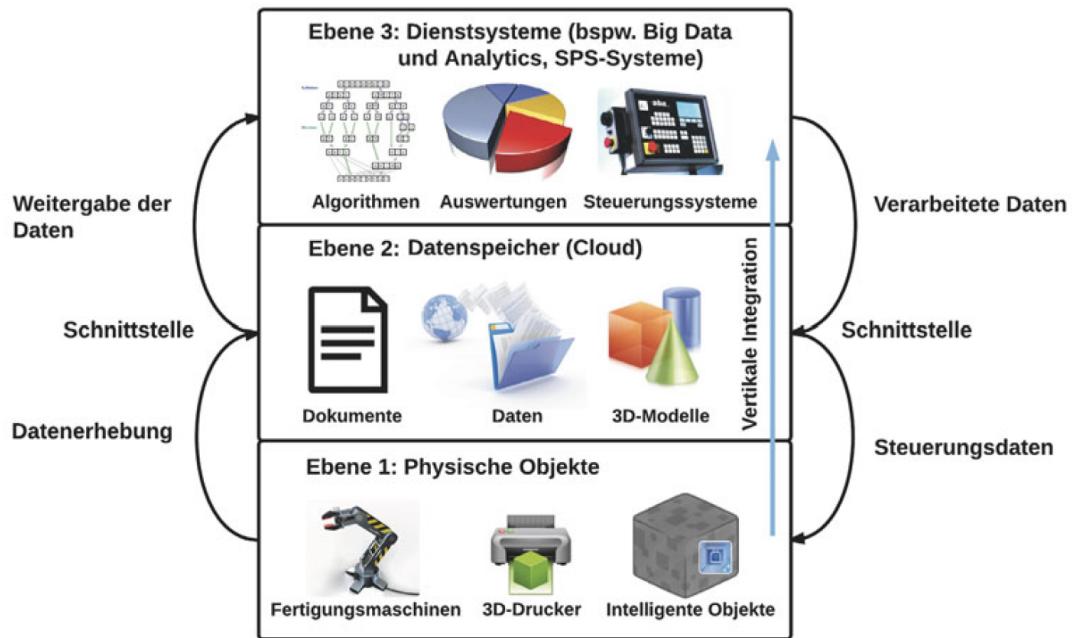


Abbildung 2: CPS in der Industrie 4.0 (Roth, 2016, S. 30)

dem Gerät gespeichert werden, sondern über das Dienste im Internet bestimmten Personen oder Parteien zur Verfügung gestellt werden: **das Internet der Dinge und Dienste** (Hänisch, 2017). Einen Mehrwert bilden die Daten durch die zentrale Aggregation und Analyse, auf deren Grundlage Entscheidungen getroffen werden können. Diese Eigenschaften lassen sich in der für IoT-Projekte typische Grundarchitektur wiederfinden: *Datentransport, Datenhaltung und Analyse* (Kenn, 2016). Der Kreis schließt sich durch die Weitergabe der verarbeiteten Daten als Steuerungsdaten an die Maschine, sodass sich der Fokus von einer Mensch-Maschinen-Interaktion auf eine Maschine-zu-Maschine-Interaktion verschiebt (s. Abbildung 2).

**Cloud Computing** stellt die IT-Infrastruktur, die ein Industrie 4.0-fähiges CPS benötigt. Der *Datentransport* vom Gerät hat oftmals eine Cloud-Plattform als Ziel, auf der die *Datenhaltung* stattfindet (Elsner et al., 2018). Die in die Cloud auslagerbaren Dienste wie Big Data oder Analytics ermöglichen *Analysen* durch Aggregationen und Auswertungen (Roth, 2016). Der Nutzen der Cloud steigert sich durch das Kombinieren der Angebote der Cloud-Dienstleister wie Amazon, SAP oder Mi-

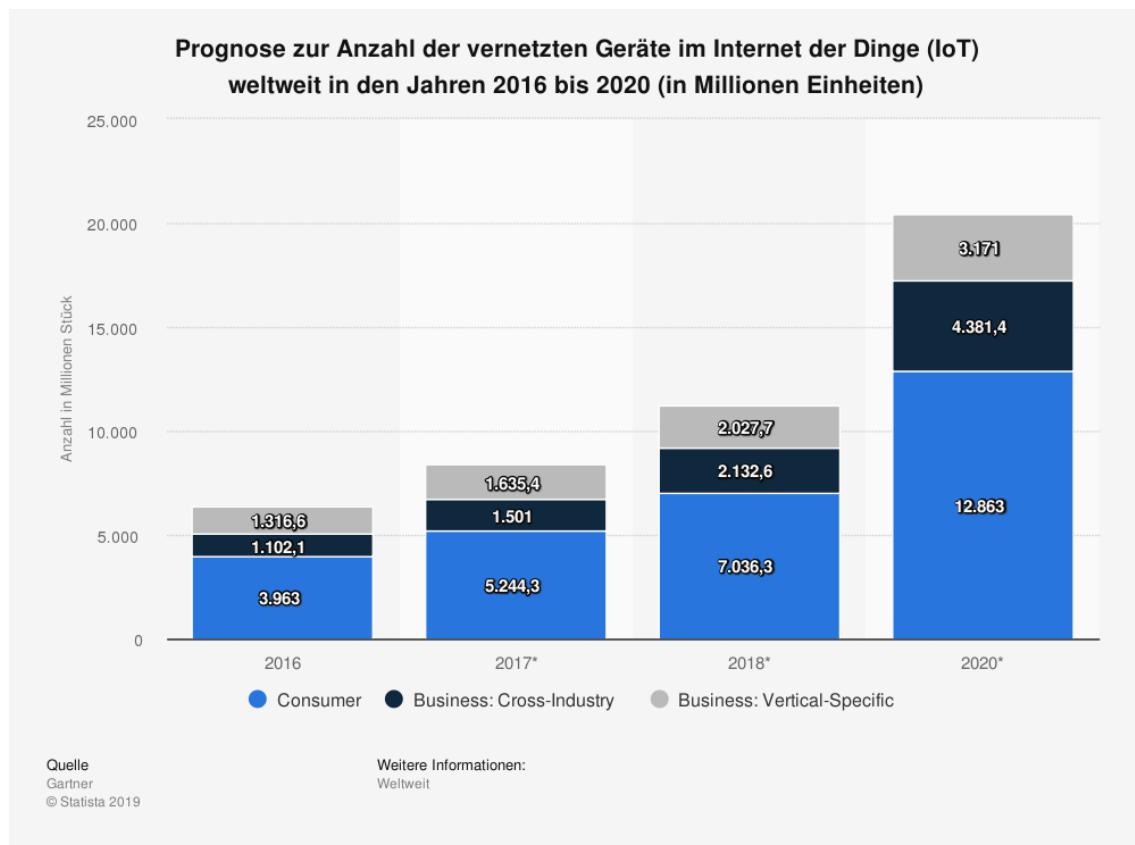


Abbildung 3: Prognose zur Anzahl der vernetzten Geräte im IoT weltweit (Gartner, 2017)

crosoft (Hänisch, 2017). Diese Anbieter stellen besitzen weltweit Rechenzentren, auf denen für Industrie 4.0 ausschlaggebende Services bereitgestellt werden:

**Big-Data-Technologien** dienen der Verarbeitung der in massiven Mengen gesammelten Daten und ermöglichen damit die Nutzung, Verwertung, Vermarktung und vor allem Analyse der digitalen Daten (Radtke und Litzel, 2019). Die Datensmengen können z.B. aus Maschine-zu-Maschine Interaktionen, Transaktionen, Verbraucherdaten oder sozialen Medien stammen (Elsner et al., 2018).

**Machine Learning** beruht sich ähnlich wie Big Data auf die *Extraktion von Wissen aus Daten* und ermöglicht dem Computer die selbstständige Ausführung bestimmter Aufgaben, ohne dass sie explizit programmiert werden müssen (Hänisch, 2017). Mittels Algorithmen können die Programme aus den Daten lernen und Muster erkennen, aus denen sie Schlussfolgerungen ziehen können (Elsner et al., 2018).

Somit können Aussagen über das wahrscheinlich zukünftige Verhalten des Systems getroffen werden (*Predictive Analytics*), aus denen in Zukunft auch Handlungsoptionen vorgeschlagen werden sollen (*Prescriptive Analytics*) (Hübschle, 2017).

**Blockchain** ist die Technologie, auf der die erste Kryptowährung Bitcoin basiert. Die Haupteigenschaften der digitalen Währung sind die Dezentralität sowie die Unveränderbarkeit und Transparenz der Transaktionshistorie. Auf Grundlage dieser Eigenschaften eines Peer-to-Peer-Systems entwickeln sich neue Geschäftsmodellmuster in verschiedensten Branchen wie Finanzen, Logistik und Transport, aber auch neue Sicherheitskonzepte (Elsner et al., 2018).

#### 2.1.4. Kommunikationssysteme

Die Vernetzung ist der Schlüssel zur Dezentralität und Autonomie der CPS - und somit zum Erfolg - in der Industrie 4.0 (Bauernhansl, 2014). 1980 stellte der Erfinder des Ethernets Robert Meltcafe eine Theorie zum Nutzen eines Netzwerks auf, die mit dem sog. *Netzwerkeffekt* in vielen wissenschaftlichen Disziplinen Anwendungen findet (Lea, 2018). Bauernhansl (2014, S. 18) beschreibt das Konzept, „[...] dass der Nutzen eines Kommunikationssystems mit dem Quadrat der Anzahl seiner Teilnehmer wächst.“ Angewandt auf das Internet of Things (IoT) stellen die Sensoren und Edge-Geräte die Nutzer dar und steigern den Wert der CPS, welche die Kommunikationssysteme bilden (Lea, 2018). Dem rasanten Anstieg der miteinander vernetzten Geräte (s. Abbildung 3) liegt außerdem das nach wie vor geltende Moore’sche Gesetz zugrunde (Barthelmä et al., 2017). Demnach verdoppelt sich die Rechner- bzw. Chipleistung alle zwei Monate bei gleichbleibenden Preisen. Folglich können Technologien, die aktuelle noch zu hohe Preise aufweisen, in Zukunft günstiger betrieben werden (Bauernhansl, 2014). Dieser Zusammenhang fördert die Fähigkeiten zur autonomen und intelligenten Vernetzung der dezentralen Systeme und führt zur Effizienzsteigerung (Barthelmä et al., 2017).

Flexibilität in der Vernetzung bieten standardisierte und simple Protokolle zum Datenaustausch im Internet wie Representational State Transfer (REST) und Message Queuing Telemetry Transport (MQTT) (Hänisch, 2017). **REST** ist ein Architekturstil, der auf dem Hypertext Transfer Protocol (HTTP) basiert und zum Lesen, Erstellen

und Bearbeiten von Ressourcen dient. Der Ansatz ist, einheitliche Anfragen an die Ressourcen-Schnittstellen per Unified Resource Identifier (URI) zu adressieren und HTTP-kodiert zu versenden (Sendler, 2016).

**MQTT** ist ein offenes Kommunikationprotokoll, welches zur Übertragung von Telemetriedaten zwischen Maschinen bei niedriger Bandbreite geeignet ist und basiert auf dem Publisher-Subscriber-Prinzip (Lea, 2018).

**Open Platform Communications Unified Architecture (OPC-UA)** ist ein industrieller Standard für den Datenaustausch im Industrie-4.0-Umfeld. Im Rahmen einer service-orientierte Architektur (SOA) ermöglichen die Schnittstellen und Protokolle die plattformunabhängige Interoperabilität zwischen den Maschinen verschiedener Hersteller (Bauernhansl, 2014).

## 2.2. Digitale Transformation mit Internet of Things

Eine Umsetzung von Industrie-4.0-Lösungen setzt technische, organisatorische und normative Bedingungen voraus. Während einige Anforderungen für alle Bereiche in der Industrie gelten, unterscheiden sich explizite Anforderungen und Lösungen je nach individuellen Ausgangssituationen von Branchen und Unternehmen (Bauer et al., 2014). Dieses Kapitel soll einen mehrdimensionalen Einstieg in die Thematik digitale Transformation bringen. Zuerst wird ein Branchenbezug für die Energiewirtschaft hergestellt. Anschließend folgen allgemeine Herausforderungen zur Umsetzung von IoT-Lösungen für die digitale Transformation. Daraufhin wird eine Referenzarchitektur zur Bewältigung dieser Herausforderungen vorgestellt. Einen wesentlichen Stellenwert in Transformationsprozessen hat das Cloud Computing als technische Voraussetzung, sodass es einer detaillierteren Erläuterung als in 2.1.3 bedarf.

### 2.2.1. Der Wandel im Energiesektor

Der Wandel in der gesamten Industrielandschaft von der Mechanisierung und Automatisierung zur Digitalisierung betrifft auf ähnliche Weise auch den Energiesektor. Es gibt in Deutschland kaum eine Branche, die sich innerhalb von zwanzig Jahren so rasant geändert hat wie die Energiewirtschaft (Doleski, 2016a). Treiber dieser schnellen Entwicklungen sind in erster Linie politisch-regulatorische Faktoren, welche für darauffolgende Anforderungen die Parameter darstellen.

Bis 1998 unterlag die Energieproduktion in Deutschland monopolistischen Strukturen. Einige wenige Versorgungsunternehmen produzierten in zentralen Werken wie Kernkraftanlagen oder Kohlekraftwerke und schleusten die gewonnene Energie in die Netze ein (Utecht und Zierau, 2018). Die Konsumenten hatten bei der Auswahl ihres Energielieferanten kaum Entscheidungsfreiheit. Mit der Liberalisierung und Privatisierung der Strommärkte öffnete sich jedoch der Binnenmarkt für den Wettbewerb (Doleski, 2017). Vor allem durch das *Unbundling*, also der Trennung von Erzeugung und Vertrieb, betraten mehrere Dienstleister für unterstützende Tätigkeiten den Markt. Auch in der Produktion stieg wegen des Wegfalls von Gebietsmonopolen der Trend von zentralen Produktionswerken zu lokalen Erzeugern in der Nähe des Verbrauchers an (Utecht und Zierau, 2018). Grundlegende Veränderungen und Innovationen kamen mit dem Ausbau von erneuerbaren Energiequellen nach der Verabschiedung des Erneuerbare-Energien-Gesetzes von 2000 zur systematischen Förderung von regenerativen Energiequellen. Vor allem nach der Nuklearkatastrophe 2011 in Fukushima wurde die Energiewende stark beschleunigt (Doleski, 2016a). Der geplante Ausstieg aus Atomkraft und Kohle zwang die Branchen, sich strukturell in der Wertschöpfungskette zu verändern. Die Herausforderung bestand vor allem darin, innerhalb der strengen gesetzlichen Regularien und eines angespannten Finanzrahmens neue Geschäftsfelder zu erschließen (Doleski, 2016a). So veränderte sich der Produktionstrend von zentraler Erzeugung zu dezentraler Erzeugung mit z.B. Windkraft und Photovoltaik. Die neuen Produktionsmechanismen bedeuten technisch gesehen einen enormen Anstieg der Steuerungskomplexität und eine Belastung der Netzinfrastruktur. Während die Produktion nun vielmehr von schwankenden Umweltbedingungen abhängig ist, bleiben die Anforderungen im Verbrauch wie die ständige Verfügbarkeit oder die stabile 50-Hz-Netzfrequenz unverändert (Utecht und Zierau, 2018).

Es sind zwar politisch-regulatorische und ökologische Faktoren, die Umwälzungen in der Energiewirtschaft erzwingen, aber die dezentrale und fluktuierende Energieerzeugung erfordert digitale Lösungen (Doleski, 2017). Der Digitalisierung wird eine Schlüsselrolle bei der Lösungsfindung für Dezentralisierung, Flexibilisierung sowie für die effiziente Nutzung von Ressourcen und Energie zugewiesen (Fraunhofer ISE, 2019). Als *Enabler* für die Energiewende konvergiert die IT-Branche immer mehr

mit energiewirtschaftlicher Leistungserstellung (Doleski, 2016a). Laut dem Bundesministerium für Wirtschaft und Energie (2015) wird die Energiebranche eine der ersten voll digitalisierten Branchen der deutschen Volkswirtschaft sein. Zwar kann der Strom nicht digitalisiert werden, aber die Vielzahl von technischen Komponenten in Anlagen müssen sowohl untereinander als auch mit dem Menschen kommunizieren können. Anders als früher ist das gesamte Stromnetz abhängig von einer Vielzahl von Erzeugungsanlagen, die ihren Strom nun in ein intelligentes Stromnetz (Smart Grid) einschleusen müssen. Das Smart Grid kombiniert in einem Netz die Erzeugung, Speicherung und den Verbrauch der Energie. Die dezentralen Erzeugungen werden durch eine zentrale Steuerung aufeinander abgestimmt, sodass die Leistungsschwankungen ausgeglichen werden können (Krone und Bachmann, 2017). Die Intelligenz wird durch den Datentransport von der Erzeugungsanlage in das Netz gewährleistet. Wenn es zu viele unkoordinierte Anlagen gibt, die ihre Produktionsmengen und ihren Zustand nicht kommunizieren können, kann es zu Instabilität im Netz führen (Umweltbundesamt, 2018). Damit die einzelnen Anlagen miteinander kommunizieren können, müssen sie Teil eines IoT-Netzwerks sein. Somit können sie ihre Umgebungs- und Zustandsdaten eigenständig an ein Condition Monitoring System (CMS) senden, das die Daten z.B. in der Cloud sammelt. Für die dort erstellten digitalen Zwillinge der realen Anlagen können auf Grundlage von Analysen der vergangenen und aktuellen Daten prädiktive Wartungsmaßnahmen abgeleitet werden. Daraus kann eine Verbesserung der Qualität und eine höhere Verfügbarkeit des Dienstes resultieren (Utecht und Zierau, 2018).

Die rasanten Entwicklungen in den letzten 20 Jahren zeigen, wie schnell sich die Branche verändern kann. Um in Zukunft auf dem Markt zu überleben, müssen sich Energieproduzenten und Dienstleister in einem ständigen Anpassungsprozess befinden (Doleski, 2016a). Essenziell für eine Anpassung an die fortwährende Vernetzung der Produktionswelt ist der Aufbau und die Umsetzung einer unternehmensspezifischen Digitalisierungsstrategie (Köster und Mache, 2017). Die Herausforderung besteht vor allem darin, die verschiedenen Disziplinen und Dimensionen in der digitalisierten Welt zu berücksichtigen (s. Abbildung 4). Da die Energiebranche besonders gesellschaftlich eine wichtige Rolle innehat, sind die Themen Datenschutz und Sicherheit nicht zu vernachlässigen (Utecht und Zierau, 2018).

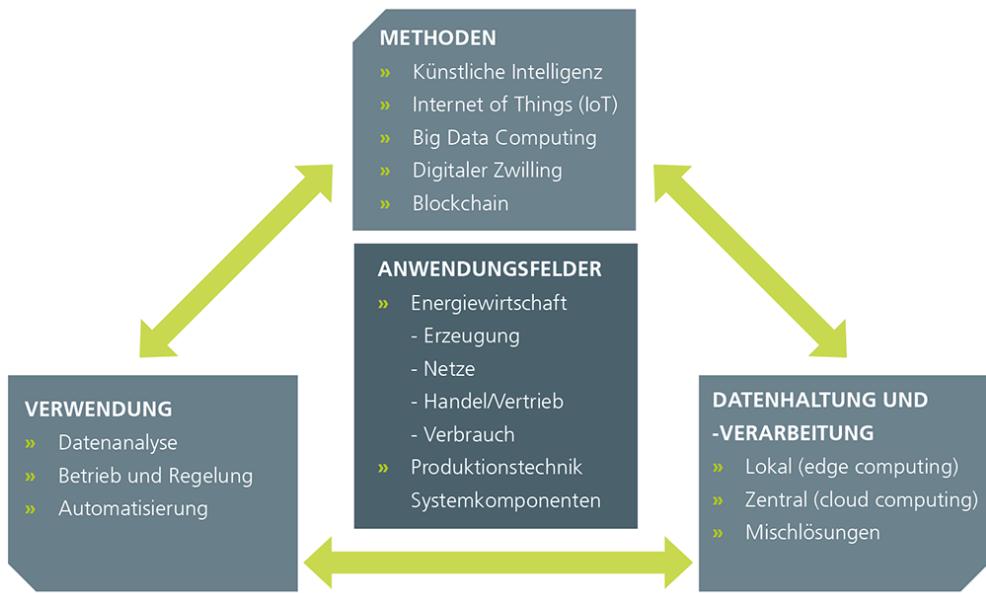


Abbildung 4: Dimensionen der Digitalisierung (Fraunhofer ISE, 2019)

### 2.2.2. Die Herausforderung IoT

Im Rahmen der Industrie 4.0, die eine Spezialisierung des Internet der Dinge und Dienste darstellt, wachsen die virtuelle und reale Welt zusammen. Daraus ergibt sich die Herausforderung, die Anforderungen der IT, der Elektrotechnik sowie des Maschinenbaus miteinander zu vereinen (Hübner, 2017).

Aufgrund der heterogenen Landschaften und Aussgangssituationen der Unternehmen sei eine Standardisierung der Technologien laut Bauer et al. (2014) unerlässlich. Des weiteren ist die Weiterentwicklung von Breitbandnetzwerken für eine echtzeitfähige Kommunikation von Systemen eine Grundvoraussetzung. Notwendig sind außerdem qualitätsgesicherte Dienste im Internet, die robust gegen Störungen sind. Der Begriff *Internet der Dinge und Dienste* bezieht auf vernetzte Komponenten wie physische Systeme, aber auch auf virtuelle Anwendungen. Da sich die Anzahl und die Beschaffenheit der Applikationen ebenso rasant ändern kann wie die der Geräte, sind eine standardisierte Laufzeitumgebung und Kommunikation für diese von großer Bedeutung. Nicht zu vernachlässigen sind dabei die Sicherheitsaspekte. Die IoT-Anwendungen bilden eine große Angriffsfläche für Hacker, die durch Sabotage und Manipulation der Systeme eine große Gefahr darstellen. Ein historisches Beispiel

für solch eine Gefahr sind die Stuxnet-Angriffe von 2010 auf iranische Atomfabriken (Bauer et al., 2014).

Allgemein können die Anforderungen und Voraussetzungen für eine IoT-Lösung auf folgende Begriffe projiziert werden (Acharya et al., 2019):

- Skalierbarkeit und Flexibilität
- Schnelligkeit
- (Ausfall-)Sicherheit
- Qualität

### 2.2.3. Vereinheitlichung durch Referenzarchitektur

Für die Bewältigung der oben aufgeführten Herausforderungen veröffentlichte die Plattform Industrie 4.0 das „Referenzarchitekturmodell Industrie 4.0“ (RAMI 4.0) sowie das Konzept zur „Industrie-4.0-Komponente“. Beide Modelle wurden 2016 nach DIN SPEC 91345 der Standardisierung zugeführt (Beuth publishing DIN, 2016).

**RAMI 4.0** ist ein branchenübergreifendes Rahmenwerk, in dem Aufgaben und Abläufe der gesamten Wertschöpfung in überschaubare Teile zerlegt und entsprechenden Normen und Standards zugeordnet werden. Das dreidimensionale Modell ist in Anlehnung auf das Smart Grid Modell erstellt und kapselt die wichtigsten Funktionalitäten aus den verschiedenen Disziplinen in Schichten. Dies schafft Flexibilität für die Konzeptionisierung und Realisierung von Industrie-4.0-Lösungen (Hübner, 2017).

Für die Migration von Produktionsgegenständen von der heutigen in die Industrie-4.0-Welt soll der gesamte Produktlebenszyklus in Daten erfasst und IT-seitig einheitlich und durchgängig abgebildet werden. Die senkrechte Achse behandelt die IT-Sicht, die die vertikale Integration der Assets in die Geschäftslogik und deren echtzeitfähige Vernetzung im Produktionsprozess beschreibt. Zu den Assets werden sowohl alle in der Anlage verbauten physischen Komponenten als auch andere Vermögensgegenstände wie Software oder Patente, aber auch Menschen, gezählt (Adolphs, 2017). Mit der Ergänzung des Assets um eine *Verwaltungsschale* entsteht die

*Industrie-4.0-Komponente*. Die Verwaltungsschale ist das Bindeglied zwischen der realen und der virtuellen Welt. Mit dessen Hilfe wird ein virtuelles Abbild des Assets samt der dazugehörigen Funktionen und Daten erzeugt. Die IT-technische Beschreibung ermöglicht die eindeutige Identifizierung des Assets z.B. durch die URI-Adresse im gesamten Wertschöpfungsprozess. Die Aktivitäten für den Übergang in die virtuelle Welt sind in der Integrationsschicht enthalten. Die Dienste zur Steuerung der Integration werden von der Kommunikationsschicht bereitgestellt. Sie dient außerdem der Vereinheitlichung der Kommunikation mit einem einheitlichen Datenformat (Bitkom e.V, 2015). Für die Datenkommunikation wird der Standard für industrielle Kommunikation OPC-UA empfohlen. Ursache dafür ist zum einen die plattformunabhängige, SOA und zum anderen die Fähigkeit, die Maschine-zu-Maschine-Kommunikation semantisch zu beschreiben. Anschließend werden die übertragenen Daten

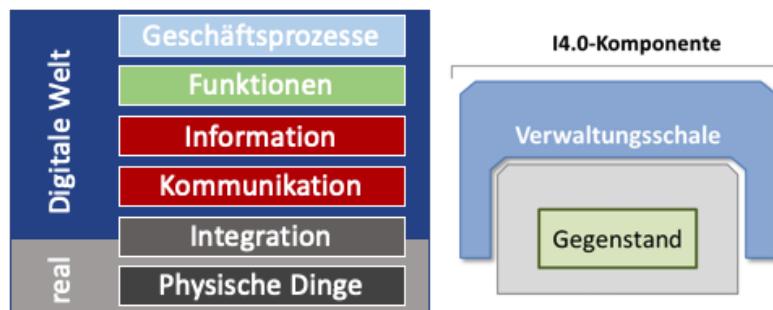


Abbildung 5: IT-sicht und Industrie-4.0-Komponente (in Anlehnung an Adolphs (2017, S. 118))

in der Informationsschicht gehalten, wo sie in einer Laufzeitumgebung in einen Regelkontext gebracht werden. Hier werden Regeln für Ereignisse und den Zugriff auf Daten definiert, die in der Funktionsschicht verarbeitet werden. Die tatsächlichen Funktionen eines Assets werden in der Funktionsschicht formal beschrieben. In der funktionalen Schicht befinden sich die Daten in einer Laufzeit- und Modellierungs-umgebung für Dienste, die unter anderem Geschäftsprozesse unterstützen. Mit dieser Plattform können die Funktionen horizontal in die Wertschöpfung integriert werden (Hübner, 2017). Zusätzlich kann in der Funktionssicht auf ERP-Funktionen zugegriffen werden. Diese entstehen in der Geschäftssicht, in der die Modellierung der Regeln für die Geschäftsprozessabwicklung stattfindet. Letztendlich werden die Assets und deren Funktionen hier in die Organisation und Geschäftsprozesse integriert.

Jede dieser IT-Schichten liegt an zwei wagerechten Achsen an, welche die horizontale Integration der Produktionsgegenstände in den gesamten Produktionprozess beschreiben. Die linke Achse soll den gesamten Lebenszyklus von Assets abbilden, während die rechte Achse diese in Hierarchiestufen der Produktions- und Automatisierungstechnik anordnet. Somit können externe Akteure wie Lieferanten oder Kunden, aber auch das erzeugte Produkt in das Industrie-4.0-Netzwerk aufgenommen werden (Bitkom e.V, 2015).

#### **2.2.4. Das neue Paradigma: Cloud Computing**

Flexibilität, Skalierbarkeit, Schnelligkeit, Sicherheit und Qualität präsentieren sich als die wichtigsten Voraussetzungen für die erfolgreiche digitale Transformation eines Unternehmens (Acharya et al., 2019). Für die Erzielung dieser Ziele spielt die in 2.1.3 bereits kurz erläuterte Cloud eine Schlüsselrolle. Denn die Technologien in der Industrie 4.0 sind zwar nicht neu, aber sie müssen in verschiedenen Kombinationen bei niedrigen Preisen und unabhängig vom Ort stets verfügbar sein. Bei alledem wird dem Nutzer je nach Bedarf das Errichten von IT-Infrastruktur und IT-Ressourcen von dem Cloud-Dienstleister abgenommen (Dzombeta et al., 2017). Entsprechend können die Cloud-Dienste aus folgenden Varianten nach nutzungsbasierten Abrechnungsmodellen wie z.B. dem Pay-Per-Use-Prinzip erworben werden:

**Infrastructure as a service (IaaS)** Bei dieser Variante stellt das Dienstleistungsunternehmen die notwendige Hardware in virtueller Form zur Verfügung. Es können je nach benötigter Menge Speicherplatz, Prozessorleistung oder Netzkapazitäten bestellt oder wieder abbestellt werden (Dzombeta et al., 2017). Kostentechnisch bietet das einen großen Vorteil, da die Server von den Anbietern angeschafft, betrieben und gewartet werden, sodass nur die verbrauchte oder vereinbarte Kapazität in Rechnung gestellt wird. Zu den global dominierenden Anbietern gehören Amazon, Microsoft und Google, die in Nordamerika, Europa und Ostasien eine hohe Dichte an Rechenzentren aufweisen (Acharya et al., 2019).

**Platform as a service (PaaS)** Das Dienstleistungsangebot dieser Variante beläuft sich auf die Bereitstellung von Middleware, Laufzeit- und Entwicklungsumgebungen

zur Erstellung von Anwendungen, Datenbanken und Webservices. Über definierte Schnittstellen (APIs) kann auf die Entwicklungsumgebung zugegriffen werden (Dzombeta et al., 2017). Da die Plattform auf IaaS basiert, fällt die Administration von Servern weg (Acharya et al., 2019).

**Software as a service (SaaS)** Kunden können bei dieser Form meist über Webbrowser auf Software(-pakete) zugreifen, die auf der Infrastruktur des Anbieters ge-hostet sind. Dabei übernimmt der Anbieter Aufgaben wie Installation, Wartung und Aktualisierung der Software (Utecht und Zierau, 2018). Dieses Prinzip ermöglicht einen schnellen Einsatz sowie eine einfache Austauschbarkeit der Software bei niedrigen Kosten. Aufgrund der Unabhängigkeit von lokalen Installationen kann die Software ortsunabhängig bei verfügbarer Internetverbindung genutzt werden (Dzombeta et al., 2017).

Die verschiedenen Modelle können auch in Kombination mit dem On-Premise-System genutzt werden.

Mit diesen Modellen bietet die Cloud einen Raum für die verschiedenen Teilsysteme, die im Industrie-4.0-Netzwerk miteinander kommunizieren und Dienste anbieten. Eine service-orientierte Architektur (SOA) ermöglicht die Interoperabilität der Akteure wie Komponentenhersteller, Automatisierer, Maschinenbauer und Softwarefirmen ohne Master-Slave-Beziehungen. In der SOA-Welt wird nicht mehr zwischen Hard- und Software unterschieden, sodass maschinelle Komponenten ihre Daten genau so als Service zur Verfügung stellen können wie eine Software ihre Funktionen bereitstellt (Adolphs, 2017). Entwicklungen wie diese verändern die grundsätzliche Denkweise in der Softwareentwicklung. Alte Architekturen werden von neuen Architekturen wie die *Microservice-Architektur* vertrieben (Acharya et al., 2019).

**Cloud-native Anwendungen und Microservices** Die Eigenschaft cloud-nativ besitzen jene Anwendungen, welche in der Cloud „geboren“ sind und durch Auschöpfung des Cloud-Potenzials die Flexibilität, Agilität und Skalierbarkeit von Cloud-Lösungen unterstützen (Acharya et al., 2019). Besonders ist dabei der agile und schnelle Entwicklungsprozess der Software. Eine cloud-native Anwendung besteht aus mehreren isolierten Services, die unabhängig voneinander entwickelt werden kön-

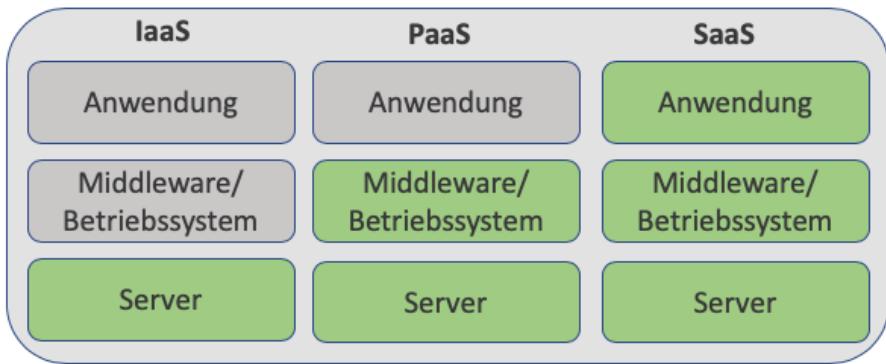


Abbildung 6: Die Cloud-Service-Modelle (In Anlehnung an Utecht und Zierau (2018, S. 88))

nen. Konventionelle Anwendungen sind im Gegensatz dazu monolithischer Natur und basieren meist auf der 3-Tier-Architektur. Mit den Bestandteilen Benutzeroberfläche, Datenbank und Anwendungsserver bilden sie ein geschlossenes System. Kleinsten Änderungen an einem der Bestandteile führen zu aufwendigen Maßnahmen zur Anpassung des Gesamtsystems (Utecht und Zierau, 2018). Anders als monolithische Architekturen zeichnet sich die Microservice-Architektur durch ihre einfache und schnelle Erweiterbarkeit und Anpassungsfähigkeit aus. Ein Microservice ist eine spezielle Funktion, oft Geschäftsfunktion, welche in Containern in einer isolierten Umgebung ausgeliefert wird und meist an eine eigene Datenbank gebunden ist. In einer Anwendung kommunizieren diese Microservices über APIs oder Messaging-Prokollen miteinander und bilden ein Gesamtsystem mit heterogenen Datenquellen. Sollte eine Funktion ein Update erfordern, muss lediglich der Microservice gewartet werden. Wenn das System eine neue Funktion erfordert, kann ein neuer Microservice einfach hinzugefügt werden, ohne Abhängigkeiten im Gesamtsystem zu stören. Infrastrukturressourcen können bei Bedarf dynamisch zu- oder abgewiesen werden (Acharya et al., 2019).

### 2.3. Werkzeuge für die Transformation

Dass die digitale Transformation durch die verschiedenen Cloud-Service-Modelle wie *IaaS*, *PaaS* und *SaaS* getrieben wird, wurde bereits erläutert. Für die Lösung der Problemstellungen dieser Arbeit sind aber nur bestimmte Produkte von Relevanz.

Aus diesem Grund wird in diesem Kapitel zunächst die technologische Basis für die Lösung, die SAP Cloud Platform, beschrieben. Daraufhin wird im Detail erläutert, was sich hinter dem Begriff *SAP Leonardo* verbirgt. Zum Schluss werden die Amazon Web Services vorgestellt.

### 2.3.1. SAP Cloud Platform

Die SAP Cloud Platform (SCP) ist eine offene PaaS, die eine Entwicklungs- und Laufzeitumgebung für cloud-native Anwendungen bereitstellt und das Rückgrat der Services des SAP Leonardo Portfolios bildet. Unterstützt wird der Betrieb von verschiedenen Global Playern unter den Infrastrukturanbietern wie Amazon Web Services (AWS), Google Cloud Platform (GCP), SAP oder Microsoft Azure. Im Grunde ermöglicht die Cloud Plattform drei Dinge. Zum einen können Anwendungen zur Erweiterung von On-Premise-Lösungen oder Cloud-Lösungen entwickelt werden. Zudem bietet SCP Integrationsservices an, um Cloud-Dienste in das SAP-Backend aber auch in heterogene Systemquellen zu integrieren. Anwendungen können aber auch mithilfe von Tools und Microservices von Grund auf entwickelt werden, ohne Hardware und Laufzeit einzurichten zu müssen (Acharya et al., 2019). Wenn von *Services* gesprochen wird, unterscheidet man zwischen zwei Arten. Die *Functional Services* werden genutzt, um Anwendungen nach eigenen Anforderungen zu entwickeln. Es werden z.B. Services aus den Bereichen User-Interface-Entwicklung, mobile Anwendungen, Sicherheit und Authentifizierung, Datenbankdienste und Integration angeboten (Elsner et al., 2018). Wenn die Services von SAP identifizierte Anwendungsfälle behandeln, werden sie als *Business Services* bezeichnet. Diese Dienste müssen für den Einsatz nur noch um den individuellen Geschäftskontext erweitert werden (Utecht und Zierau, 2018).

Wie die Anwendungen und Services entwickelt und verwaltet werden können, hängt von der Umgebung ab. Die SCP stellt drei Umgebungen zur Verfügung, von denen jede mindestens eine Laufzeitumgebung sowie verschiedene Tools zur Entwicklung bereitstellt: Neo, Cloud Foundry und ABAP.

**Neo-Environment** Die SAP Cloud Platform wurde 2012 erstmalig nur in der Neo-Umgebung angeboten und unterstützte nur die Entwicklung von Java-, nativen HANA- und HTML5-Anwendungen (Elsner et al., 2018).

**Cloud Foundry-Environment** 2016 wurde die Beta-Version der Open-Source Multi-Cloud-PaaS von VMWare und General Electric entwickelt (Utecht und Zierau, 2018). Schließlich haben sich viele große Technologieunternehmen wie IBM, Pivotal, Google oder SAP sich für eine gemeinsame Laufzeitumgebung entschieden. Somit wurden sie Teil der Non-Profit-Initiative *Cloud Foundry Foundation*, um die Technologie weiterzuentwickeln. Durch die gemeinsame Basis wird die Interoperabilität von verschiedenen Programmiersprachen und Applikationsdiensten gewährleistet (s. Anhang ??). Deshalb ergänzte SAP die SCP um die Cloud Foundry-Umgebung, in der die Nutzung von Services für das Internet der Dinge, Blockchain oder Machine Learning möglich ist (Elsner et al., 2018). Aufgrund der Kollaboration und den daraus entstandenen Möglichkeiten ist Cloud Foundry mittlerweile ein industrieller Standard geworden (Acharya et al., 2019).

### 2.3.2. SAP Leonardo

Die SAP Cloud Platform bildet die technologische Grundlage für die digitale Transformation mit SAP. Diese Technologien werden um die *SAP-Leonardo-Technologien* ergänzt. Der Begriff lehnt sich an den Innovationsgeist Leonardo da Vincis an und soll die *digitale Renaissance* mit SAP darstellen (Howells, 2017). Es gilt zu betonen, dass SAP Leonardo nicht als Plattform oder Produkt zu verstehen ist. Es handelt sich vielmehr um eine Sammlung von Functional Services (Vgl. 2.2.4 Microservices) und Anwendungen der SCP, die als zukunftsgestaltende Technologien gelten (Elsner et al., 2018). Zu diesen Technologien gehören neben dem Internet der Dinge (Internet of Things (IoT)) auch Machine Learning, Big Data, Blockchain und Advanced Analytics. Wie bereits zuvor erwähnt, sind diese Technologien in ihrem Wesen nicht neu. Der innovative Charakter entsteht erst durch die Fähigkeit, die zusammenhängenden Technologien miteinander zu verknüpfen (Utecht und Zierau, 2018). Für den Zweck bietet SAP vierlei Lösungen:

Einerseits haben Kunden die Möglichkeit, fertige Lösungen *Powered by SAP Leonardo* zu erwerben, welche nur noch an die Anforderungen des Unternehmens angepasst werden müssen (Utecht und Zierau, 2018). Beispiele für solche Anwendungen sind *Connected Goods*, *Connected Assets* oder *Connected Infrastructure* (Elsner et al., 2018). Mit der Cloud-Anwendung *SAP Leonardo Bridge* können die Daten aus die-

sen IoT-Anwendungen mit den Backend-Daten zugesammengeführt, dargestellt und verarbeitet werden. Das Produkt IoT-Edge ist die Schnittstelle zwischen den Sensoren am Gerät und der Cloud. Bevor die Daten an die Cloud übertragen werden, können sie im Offline-Betrieb gepuffert, aggregiert und vorverarbeitet werden (Utecht und Zierau, 2018).

Allerdings können IoT-Szenarien mit der *SAP Leonardo IoT Foundation* agil selbst entwickelt werden (Elsner et al., 2018). Hierfür ist eine Sammlung von Microservices und Application Programming Interface (API) auf der SCP bereitgestellt. Diese können für das *Device Management* und *SAP IoT Application Enablement* verwendet werden. Im *Device Management* werden die Geräte nach einem vorgedachten Modell verwaltet, für die im *Application Enablement* ein *digitaler Zwilling* als Grundlage für die Anwendungsentwicklung erstellt wird (Elsner et al., 2018).

Im Grunde ist SAP Leonardo ein Sammelbegriff für alle IoT-relevanten Produkte (Utecht und Zierau, 2018). Kennzeichnend für die Innovation ist neben der Interoperabilität der Services die Interoperabilität der Unternehmensdaten. Das Unternehmensgedächtnis ist in dem digitalen Kern, also dem Enterprise-Resource-Planning (ERP)-System, verankert (Elsner et al., 2018). Der digitale Kern kann nahtlos mit der agilen Entwicklung auf dem *Digital Innovation System* verbunden werden. Mithilfe z.B von SAP zur Verfügung gestellten Design Thinking Methoden können aus kontextbezogenen Stamm- und Historiendaten des Unternehmens Erkenntnisse für Geschäftsprozesse geschlossen werden (Elsner et al., 2018).

SAP spricht von einem Ökosystem, in dem Entwicklungs- und Implementierungs-partner mit den Technologien interagieren, um innovative und effiziente Produkte zu erzeugen. Während Implementierungs-partner dabei helfen, bereits bestehende Lösungen und Anwendungen zu konfigurieren und anzupassen, entwickeln die Entwicklungs-partner auf Basis der SAP-Leonardo-Technologien neue Anwendungen. Dies ist vor allem im Kontext der Verschmelzung der IT mit dem Maschinenbau und der Elektrotechnik interessant. Die Konvergenz der Themenfelder bewegt reine Hardwarehersteller zu der Entwicklung von intelligenten Softwarelösungen als digitale Service Provider (Elsner et al., 2018). Das Ökosystem beinhaltet zudem die in Abschnitt 2.3.1 erwähnten Infrastrukturpartner, Softwaredienste und Laufzeitumgebungen aber auch z.B. Standards und Normen.

### **2.3.3. Amazon Web Services**

Amazon Web Services (AWS) ist ein Cloud-Computing Anbieter aus den USA und ist eine Tochtergesellschaft des Online-Versandhändlers Amazon. Mit 35 % Marktanteil ist das Unternehmen international führender Cloud-Dienstleister sowohl für Unternehmen als auch für Privatpersonen (mind square, 2019). Mit Rechenzentren in den USA, Europa, Brasilien, Asien und Australien stellt es die größte Public-Cloud der Welt zur Verfügung. Zu den Dienstleistungen gehören unter anderem die PaaS Elastic Beanstalk als Entwicklungsumgebung, Anwendungen, Datenbank- und Speicherdiene sowie Netzwerke. Im AWS Marketplace können sowohl die Services von AWS als auch von Dritten bereitgestellte Services gebucht werden. Einige bekannte Dienste von AWS sind der Simple Notification Service (SNS) oder der Simple Workflow Service. In erster Linie stellt AWS die Infrastruktur und Rechenleistung zur Verfügung, die z.B. auch nur für das Hosten eigener Websites genutzt werden kann (AWS, 2019). Das Unternehmen stellt jedoch auch die Infrastruktur für PaaS anderer großer Firmen wie SAP, mit denen seit 2011 eine Infrastrukturpartnerschaft besteht (Elsner et al., 2018). Sowohl die SAP Cloud Plattform sowie einzelne Services wie die der SAP Leonardo IoT Foundation laufen standardmäßig auf Rechenzentren von AWS. Weitere bekannte Großkunden sind Netflix, die NASA oder Dropbox (mind square, 2019).

### **3. Umsetzungskonzept für die digitale Transformation**

Für die Beantwortung der Forschungsfragen (s. Kapitel 1.2) wurde im Kapitel 1.3 ein Lösungsansatz bereits vorgestellt. In dem folgendem Kapitel wird die Umsetzung des Lösungsansatzes dargestellt. Zuerst wird ein repräsentativer Anwendungsfall für die Energiewirtschaft entwickelt. Für die Spezifikation der Zielarchitektur des Prototypen wird vorher eine detaillierte Anforderungsanalyse durchgeführt. Außerdem wird vorher eine Systemanalyse der zugrundeliegenden Systemarchitektur von SAP Leonardo durchgeführt. Auf Basis dieser Schritte wird schließlich die das System des Prototypen entworfen und umgesetzt.

#### **3.1. Repräsentativer Anwendungsfall für die Energiewirtschaft**

Die verschiedenen Werttreiber und Anforderungen für ein Digitalisierungskonzept unterscheiden sich je nach Unternehmen und Branche. Für eine erfolgreiche Transformation müssen daher individuelle Anwendungsfälle identifiziert werden. In Anbe tracht der Dynamik und des rasanten Tempos, in der neue Technologien entstehen, ermöglicht ein anwendungsfallbasierter Ansatz eine flexible und agile Anpassung. (Acharya et al., 2019, S. 31)

Aus diesen Gründen wird im Folgenden ein repräsentativer Anwendungsfall für die Energiebranche vorgestellt. Die Anforderungen an das Zielsystem werden nach den von Lauenroth et al. (2016) vorgestellten Methoden erhoben.

##### **3.1.1. Ausgangsszenario**

Der Windenergieanlagenhersteller Enercon GmbH aus Aurich verzeichnet 29000 Anlagen in 45 Ländern. Da das Kerngeschäft des Unternehmens auf den Bau von Anlagen für die dezentrale Energieerzeugung basiert, hat Industrie-4.0-Fähigkeit einen besonderen Stellenwert. Sei es die Einspeisung der produzierten Energie in das Smart-Grid, die Fernsteuerung oder die Zustandsüberwachung der Anlagen und Windparks: Das unternehmenseigene Supervisory Control and Data Acquisition (SCADA)-System ist auf die Enercon-Anlagen abgestimmt und bietet umfangreiche Lösungen für die Kunden. Allerdings versendet das SCADA-System die Messwerte bisher nur alle 15 Minuten das CMS. Zudem ist es eine technische Insellösung,

welche die betriebswirtschaftliche Welt nicht integriert. Als Kunden stehen die Energieversorgungsunternehmen im Vordergrund, mit denen Enercon Verträge für Wartungsservices abschließt. Die Enercon-IT nutzt SAP-Produkte für das Management der Ressourcen, Logistik oder Kunden. Im Zuge der Anpassung an die Anforderungen der digitalen Welt wird SAP jedoch den Support der bisher auch von Enercon verwendeten Standard-ERP-Software bis 2025 einstellen. Der Fokus wird auf das Nachfolgeprodukt SAP S/4 HANA gesetzt, welche die echtzeitfähige In-Memory-Datenbanktechnologie High Performance Analytic Appliance (HANA) für nutzt. Aus diesem Grund bereitet sich Enercon rechtzeitig auf die Migration auf S/4 HANA vor. Die Integrations- und Entwicklungsplattform von HANA bietet zahlreiche Möglichkeiten zur Realisierung von innovativen Softwarelösungen sowohl auf der Cloud als auch On-Premise. Die Neuausrichtung der IT-Architektur ist auch für die erfolgreiche digitale Transformation von Enercon großer Bedeutung.

Im besonderen Interesse liegt die SAP Leonardo IoT Foundation, vor allem in Anbetracht einer möglichen Integration von Stammdaten aus dem S/4 HANA System. Dafür ist zunächst eine Analyse der SAP Leonardo Systemarchitektur mitsamt der Perspektiven gewünscht. Dies könnte als Entscheidungsgrundlage für eine Erweiterung des Geschäftsfeldes von Energieproduktion auf IT-Dienstleistungen dienen. Außerdem soll prototypisch dargestellt werden, inwiefern sich SAP Leonardo IoT als Verwaltungsschale für die Industrie-4.0-Komponente eignet. Langfristiges Ziel des Unternehmens sei es, das SCADA-System echtzeitfähig zu gestalten. Um Risiken vor Inbetriebnahme und Kosten zu minimieren soll jedoch zunächst eine einfache Simulation genügen.

Die Simulation soll dem Servicepersonal in der Wartung und den Kunden ermöglichen, die Zustandsdaten des digitalen Zwillings einer Anlage in Echtzeit zu überwachen. Wenn kritische Messwerte empfangen werden, soll das Personal sofort benachrichtigt werden, damit Wartungsmaßnahmen eingeleitet werden können. Die Softwareentwickler/innen sollen den Prototypen beliebig sowohl um (Mess-)Geräte als auch um App-Funktionalitäten erweitern können.

### **3.1.2. Anforderungserhebung**

Um die Anforderungen für die Umsetzung einer repräsentativen Lösung zu bestimmen, muss zunächst ermittelt werden, welche Einflussfaktoren sich im Kontext des Zielsystems befinden und wo sich die Grenze des Systems befindet. Mit der Evaluation der Ausgangssituation (s. 3.1.1) können Anforderungsquellen identifiziert werden, welche sich auf das Zielsystem beziehen und sich im Systemkontext befinden. Auf Grundlage der Evaluation werden zunächst Probleme, Anforderungen und Lösungen für das System definiert. Für eine bessere Strukturierung des Systems werden die einzelnen Problem-Anforderung-Lösung (PAL) auf die Ebenen System und Systemkontext, sowie auch auf die technische Ebene abstrahiert. Die Dokumentation von Anforderungen auf verschiedenen Abstraktionsebenen ist vor allem für die nachträgliche Verwaltung der Anforderungen für zukünftige Softwareprojekte von großem Wert (Lauenroth et al., 2016). Nach IREB e.V. (2017) unterscheidet man typischerweise zwischen drei Arten von Anforderungen: funktionale Anforderungen, Qualitätsanforderungen sowie Randbedingungen. Die Formulierung von Anforderungen setzt das Vorhandensein einer Lösung voraus, weshalb auch der Prototyp bereits als Anforderungsquelle gelten kann.

Aus der Evaluation der Ausgangssituation ergibt sich die in Abbildung 7 dargestellte Systemabgrenzung. Die Anforderungsanalyse berücksichtigt die Stakeholder, welche sowohl mit dem System interagieren als auch direkten Einfluss auf die Anforderungen haben. Als Nutzer des Systems kristallisieren sich die Kunden Enercons (z.B. die Energieversorgungsunternehmen (EVU)) sowie das Servicepersonal in der Wartung. Stakeholder mit qualitativen Anforderungen an das System sind die Softwareentwickler/innen und -architekt(inn)en, für welche der Prototyp als Architekturvorlage dienen soll. Die Auftraggeber stellen die direkte Anforderung, eine Lösung und Analyse für SAP Leonardo zu entwickeln, die die Eignung SAP Leonards als Verwaltungsschale für die Industrie-4.0-Komponente (I-4.0-K) beurteilt. Dies impliziert, dass sich an der IT-Sicht (s. Abbildung 5) der Referenzarchitektur RAMI 4.0 und an dem Konzept der Industrie-4.0-Komponente (s. Abschnitt 2.2.3) orientiert werden soll. Berücksichtigt werden sollen außerdem die Anforderungen, die sich aus der Beschaffenheit der Branche der Energiewirtschaft ergeben. Bereits

existierende Systeme können ebenfalls Einfluss auf das System haben, was jedoch für die Prototypentwicklung nicht relevant ist.

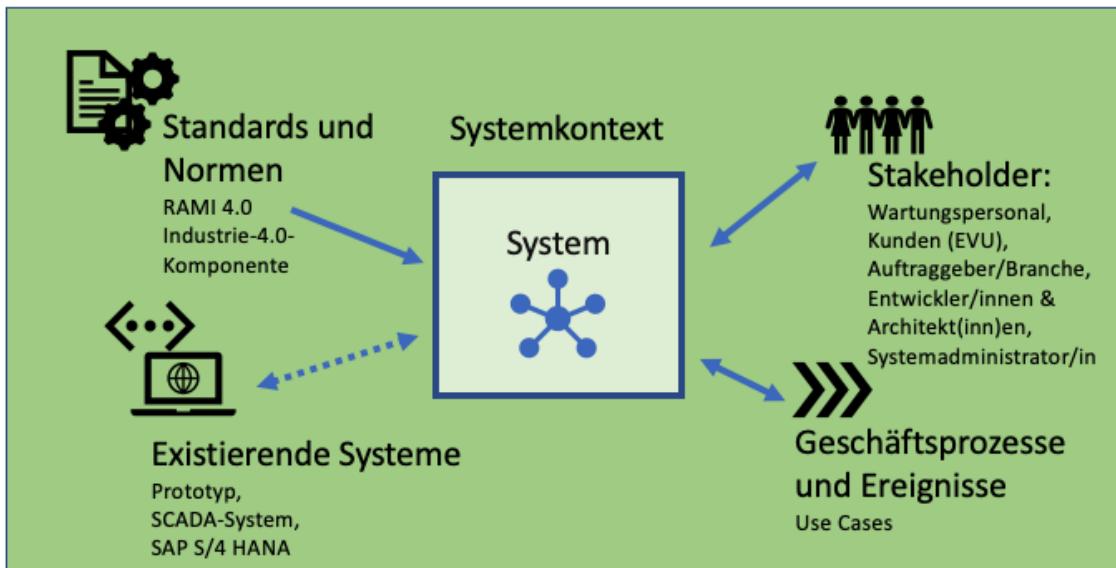


Abbildung 7: Systemabgrenzung und Systemkontext

### 3.2. Anforderungsanalyse

Im folgenden Kapitel wird die Anforderungsanalyse nach Abstraktionsebenen des PAL-Modells (s. Tabelle 1) durchgeführt. In der Kontextebene werden Anforderungen bestimmt, welche sich direkt oder indirekt die Funktionen des Systems bestimmen. Die Behandlung der Forschungsfrage FF1.1, welche Anforderungen an ein System für die digitale Transformation sich aus Sicht der dezentralen Energierzeugung ergeben, schafft die Grundlage für die Problemdefinition für die Funktionsweise des eigentlichen Systems. Die Funktionen werden in der Systemebene mit den notwendigen Schnittstellen und Datenstrukturen in einen logischen Aufbau eingeordnet. Anschließend wird auf der technischen Ebene der logische Aufbau technisch beschrieben.

#### 3.2.1. Kontextebene

**Problemstellungen** Die Probleme, die durch das Zielsystem gelöst werden sollen, sind durch verschiedene Einflussfaktoren aus dem Kontext des Systems verursacht.

Problem	Anforderung	Lösung
<b>Kontextebene</b>		
K-P-N	K-A-N	K-L-N
<b>Systemebene</b>		
S-P-N	S-A-N	S-L-N
<b>Technische Ebene</b>		
T-P-N	T-A-N	T-L-N

Tabelle 1: Das PAL-Modell

In erster Linie steht das Problem der dezentralen Energieerzeugung aus dem Branchenkontext (s. Abschnitt 2.2.1). Da die Erzeugung von schwankenden (Umwelt-) Bedingungen abhängt, müssen kontinuierlich Daten erhoben werden, um Leistungsqualität und -verfügbarkeit zu gewährleisten. Gleichzeitig steigt der Koordinationsaufwand aufgrund der großen Datenmengen. Weil die SCADA-Systeme Messdaten nur im 15-Minuten-Takt versenden, können keine aktuellen Zustandsdaten eingesehen und nicht rechtzeitig auf Probleme reagiert werden. Problematisch ist dies besonders in Anbetracht der erhöhten Steuerungskomplexität der Anlagen. Als ein weiteres Problem kann die hohe Abhängigkeit der Branche von gesetzlichen Vorgaben gesehen werden. Die sich regelmäßig ändernden Regularien können die Strukturen und die Beschaffenheit der Branche grundlegend ändern. Aus diesem Grund kann sich die Umsetzung von ohnehin schon komplexen und interdisziplinären Industrie-4.0-Projekten für die Energiebranche als große Herausforderung erweisen, aber auch einen enormen Mehrwert bringen. Zudem ergibt sich aus dem Ausgangsszenario die Problematik, dass das bestehende System zur Zustandsüberwachung keine Integration von intelligenten Diensten ermöglicht. Mit dem Umstieg auf SAP S/4 HANA stellt sich die Frage, inwiefern sich das Innovationsportfolio SAP Leonardo als Verwaltungsschale für die Anlagen eignet. In diesem Zusammenhang ergibt sich aus dem Ausgangsszenario jedoch die Problematik des erhöhten Risikos bei großen Industrie-4.0-Projekten. Auch Lauenroth et al. (2016) mahnen bei Projekten für Anlagen mit komplexer Systemelektronik und -mechanik zur Vorsicht. Ein Change Request für solch komplexe Systeme wie Windenergieanlagen wäre zu teuer.

ID	Problem	Quelle
<b>K-P-1</b>	Anstieg der Steuerungskomplexität der Anlagen wegen der dezentralen Energieerzeugung	<i>Branche</i>
K-P-1.1	Koordination großer Datenmengen	
K-P-1.2	Die Gewährleistung der Leistungsqualität und Leistungsverfügbarkeit	
K-P-1.3	Verzögerung der Reaktion auf Probleme aufgrund des 10-Minuten-Takts der SCADA-Systeme	<i>Auftraggeber</i>
<b>K-P-2</b>	Strenge Regularien können die Branche stetig ändern	<i>Branche</i>
<b>K-P-3</b>	Interdisziplinäre und komplexe Struktur von Industrie-4.0-Projekten	<i>RAMI 4.0</i>
<b>K-P-4</b>	Eignung der SAP Leonardo Foundation als Verwaltungsschale	<i>Auftraggeber</i>
K-P-4.1	Nutzung von intelligenten Diensten	
K-P-4.2	Erhöhtes Risiko bei der Umsetzung großen Industrie-4.0-Projekte	

Tabelle 3: Probleme aus Kontextebene

**Kontextmodell** Die oben definierten Probleme schaffen eine Struktur für ihre Lösung (vgl. PAL-Modell Lösungssäule). Mit dem Kontextmodell wird eine erste statische Struktur des Zielsystems auf Grundlage der identifizierten Anwendungsfälle und Lösungsmöglichkeiten geschaffen (Lauenroth et al., 2016). Als Nutzer des Systems bestimmen die Anwendungsfälle des Kunden und des Wartungspersonals die erste grobe Struktur des Systems, doch sie wird genau so durch die Rahmenbedingungen im Systemkontext geformt. Mit dem in Abbildung 8 dargestellten Use Case Diagramm werden zusammenhängende Anwendungsfälle zur Lösung der Probleme in K-P-1 aufgeführt. In dem Diagramm wird der gewünschte Geschäftsprozess des Auftraggebers für die Nutzer in Elemente aufgeteilt, die durch das System aufgegriffen werden.

Gelöst werden die Probleme jedoch in erster Linie durch die Verfügbarkeit einer

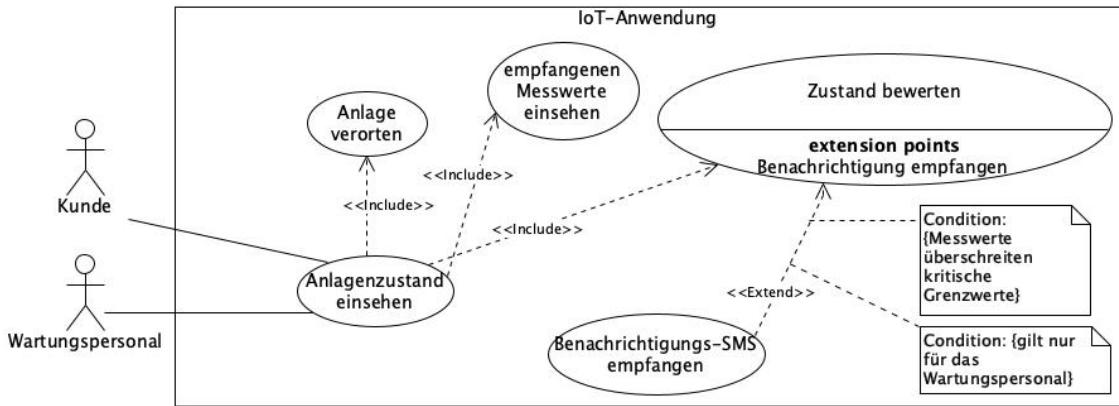


Abbildung 8: Use Case Diagramm der Kontextebene

intelligenten Verwaltungsschale über der physischen Anlage nach dem Konzept der Industrie-4.0-Komponente (s. Abschnitt 2.2.3). Ein wesentliches Merkmal von Industrie-4.0-Projekten ist die Interdisziplinarität und Komplexität (K-P-3). Für den Aufbau einer Strategie und die Bewältigung der Herausforderungen, die ein komplexes System stellt, wird eine von Politik und Wirtschaft entwickelte Referenzarchitektur als Hilfe herangezogen. Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0) bildet (s. Abschnitt 2.2.3) eine wichtige Entscheidungsgrundlage für die Beurteilung der Eignung (K-P-4) der prototypischen Architektur eines solchen Systems. Da die Umsetzungsstrategie der Plattform Industrie 4.0 (Bitkom e.V, 2015) erneuerbare Energien nicht berücksichtigt, aber die Abhängigkeit von fluktuierenden Regularien (K-P-2) besteht, soll eine unternehmensspezifische Architekturvorlage entwickelt werden.

**Anforderungen** Die Anforderungsdefinition in der Kontextebene ist von hoher Abstraktion geprägt. Sie entstehen aus den wesentlichen Problemen, die das Zielsystem zu lösen hat. Es wird zunächst ein grober Überblick über die wichtigsten Anforderungen an das Gesamtsystem gegeben, um eine Orientierung für die Anforderungserhebung auf Systemebene zu schaffen. Nach Doleski (2016b) gibt es drei wesentliche Herausforderungen für Energieunternehmen. Zum einen gilt es die Informationsflut aus der dezentralen Produktion zu bewältigen. Bezogen auf das Zielsystem bedeutet dies, Messwerte aus der Anlage in einem digitalen Zwilling visuell bereitzustellen. Außerdem müssen diese Informationen Wissen erzeugen. Das System muss

dies durch die Bereitstellung von prädiktiven Informationen und durch Reaktion auf kritische Zustände ermöglichen. Zudem müssen aus den Informationen relevante Erkenntnisse für die Unternehmensführung gewonnen werden. Dafür muss das Zielsystem eine Möglichkeit für die Einbindung von intelligenten Diensten zur Datenverarbeitung aufweisen. Für einen besseren Überblick sind die Anforderungen im Anhang A.1 gelistet.

### 3.2.2. Systemebene

Um den inneren logischen Aufbau Prototypen zu spezifizieren, müssen die Lösungen und Anforderungen der Kontextebene erneut in zu lösende Probleme zerlegt werden. Anschließend wird das Systemmodell vorgestellt, welches mit seinen Schnittstellen, Funktionen und Datenstrukturen die Probleme lösen soll. Da aus der Spezifikation der Systemfunktionen sich neue Anwendungsfälle ergeben, werden diese ebenfalls spezifiziert. Schließlich werden die Anforderungen an das System definiert.

**Problemstellungen** Damit der Prototyp die Anforderungen der des Systemkontextes lösen kann, muss es spezielle Probleme lösen. Das Hauptproblem ist die Virtualisierung eines physischen Assets mitsamt der zugehörigen Messdaten. Unabhängig davon, wo sich der Nutzer befindet, soll er jederzeit den Zustand der Anlage durch den digitalen Zwilling einsehen können. Dabei soll es dem Nutzer ermöglicht werden, den digitalen Zwilling eindeutig einer realen Anlage zuzuordnen. Um einen Mehrwert aus den gemessenen Daten zu erlangen, muss der Prototyp die Bedeutung bestimmter Daten erkennen und ggf. eine Benachrichtigung versenden.

**Systemmodell** Damit der Prototyp die Anwendungsfälle aus Abbildung 8 ermöglichen kann, enthält es im Inneren bestimmte Funktionen, Schnittstellen und Datenstrukturen. Von der Virtualisierung der Anlage bis zur Präsentation für den Nutzer fließen viele Daten. Der Fluss der Daten über die Funktionen und Schnittstellen ergibt ein erstes grobes Systemmodell (s. Abbildung 9). Die gelben Objekte repräsentieren sind Schnittstellen<sup>1</sup>, die die Verbindung des Systems mit der Umwelt beschreiben. Über die Funktionen (grün) fließen die Daten an die Datenspeicher

---

<sup>1</sup>Die Notation ist an das Datenflussdiagramm angelehnt, wobei die Darstellung leicht verändert wurde.

ID	Problem	Quelle
<b>S-P-1</b>	Übergabe des pyhsischen Assets in die digitale Welt	<i>K-FA-1</i>
S-P-1.1	Empfang aller Zustandsdaten der Anlage	
S-P-1.2	Identifikation der Anlage	
S-P-1.3	Erkennung der Bedeutung eines Messwerts	<i>K-FA-1.3</i>
S-P-1.4	Orts- und zeitunabhängige Anzeige der Zustandsdaten einer Anlage	
S-P-1.5	Verortung einer Anlage	<i>K-FA.1.2</i>
S-P-1.6	Erkennung der Grenzüberschreitung eines Messwerts	<i>K-FA-1.4</i>
S-P1.7	Erkennung der Notwendigkeit einer erneuten Benachrichtigung	
<b>S-P-2</b>	Bereitstellung einer flexiblen Systemarchitektur	<i>K-QA</i>
<b>S-P-3</b>	Bereitstellung einer standard-konformen Systemarchitektur	<i>K-RA</i>

Tabelle 5: Probleme aus Systemebene

(rot), bis sie schließlich bei den Nutzern ankommen. Dieses Diagramm dient nicht dazu, den Datenfluss zu kontrollieren, d.h. Bedingungen zu setzen. Es soll lediglich einen Überblick über den Aufbau des Systems geben.

Aus dem Diagramm sind die **Schnittstellen** Anlage, Systemadmin, die Nutzer und der Benachrichtigungsdienst zu entnehmen. Technisch gesehen existieren noch weitere Schnittstellen, über die Daten verarbeitet und ausgetauscht werden.

1. *Systemadmin* zur *Cloud*: Der Systemadmin soll über Benutzerschnittstellen zur Geräteverwaltung Typen und Instanzen der Anlage definieren. Eine Anlage soll aus n Sensoren und einem Gateway bestehen.
2. *Sensoren* zur *Anlage*: Sensoren sollen Windgeschwindigkeit (km/h), Luftfeuchtigkeit (%), Temperatur (°C), Luftdruck (hpA) und die Luftpumpe ( $m^3$ ) erfassen.
3. *Anlage* zur *Cloud*: Messwerte sollen von der Anlage über ein *Gateway* an die Zieladresse der Anlageninstanz in der Geräteverwaltung gesendet werden.
4. *Systemadmin* zum *digitalen Zwilling*: Der Systemadmin soll über Programmierschnittstellen Anlagentypen und -instanzen für die Zwillinge erzeugen,

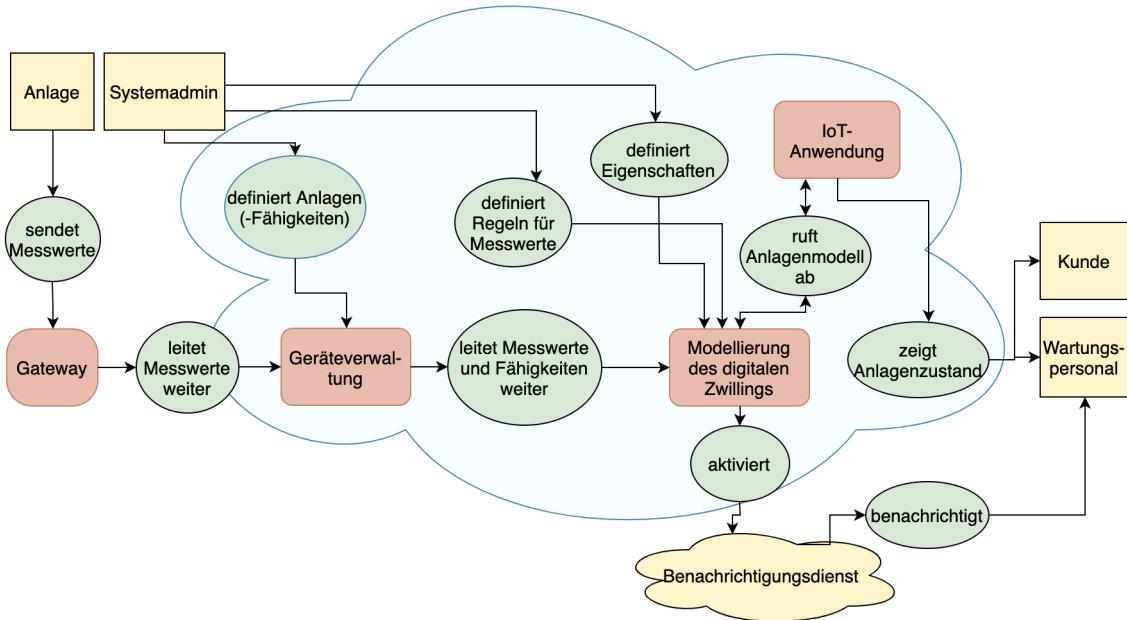


Abbildung 9: Datenflussdiagramm

damit sie in eine Anwendung integriert werden können. Außerdem sollen Regeln und Funktionen für die Anlagentypen definiert werden.

5. *Anlageninstanz zum digitalen Zwilling:* Die Messwerte der Anlage sollen dem digitalen Zwilling übergeben werden.
6. *Benachrichtigungsdienst zum digitalen Zwilling:* Der Benachrichtigungsdienst soll über die definierten Regeln für die Anlage aktiviert werden.
7. *Nutzer zur Anwendung:* Die Nutzer sollen die Anlagen über eine grafische Benutzerschnittstelle einsehen können.
8. *optional: Digitaler Zwilling zur Anlage:* Der digitale Zwilling soll Befehle an die Anlage senden.

**Anwendungsfälle** Damit die Anwendungsfälle der Nutzer erfüllt werden können, müssen innerhalb des Systems neue Anwendungsfälle behandelt werden. Die neuen Fälle betreffen die Funktionen, die der Systemadministrator ausführen muss, damit die physische Anlage und dessen Daten virtuell repräsentiert werden können. In

Abbildung 10 werden die neu ermittelten Anwendungsfälle in einem neuen Use Case Diagramm dargestellt.

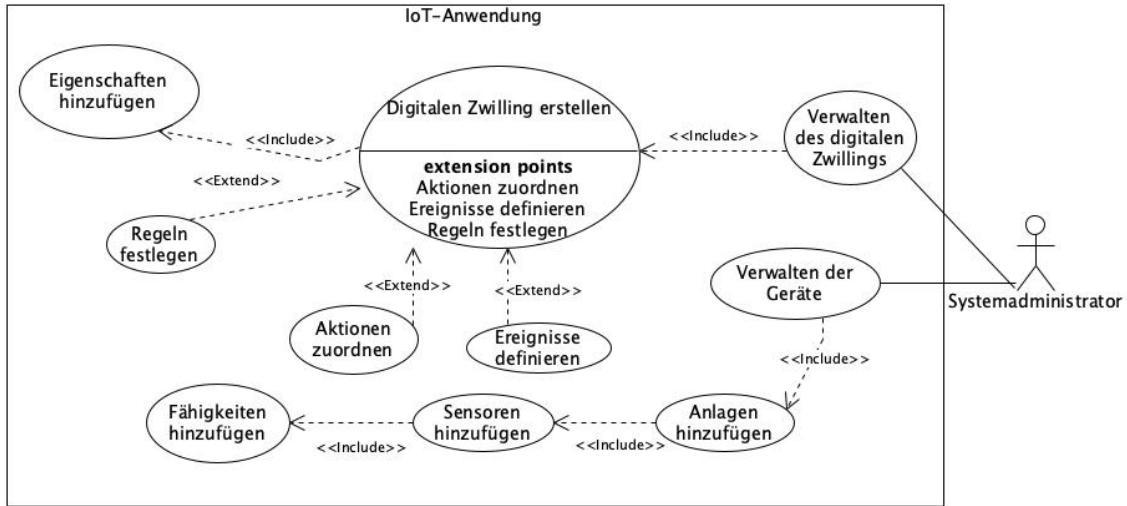


Abbildung 10: Erweitertes Use Case Diagramm auf Systemebene

**Anforderungen** Die Evaluation des Lösungsmodells mit dem Anwendungsfalldiagramm und dem Datenflussdiagramm ergibt die wesentlichen Teilsysteme des geplanten Systems. Die funktionale Anforderungsdefinition bezieht sich demnach auf die Teilbereiche, die ein Gesamtsystem ergeben (s. Anhang A.3):

1. *Das Messinstrument zur Simulation einer Anlage* (S-FA-1) muss min. alle 5 Sekunden die Daten erfassen, verarbeiten und direkt an die Geräteverwaltung senden können.
2. *Die Integration und Virtualisierung der Anlage* (S-FA-2) einschließlich der Behandlung der Anwendungsfälle muss vom Prototypen gewährleistet werden.
3. *Die Visualisierung der Anlage* (S-FA-3) muss durch eine Benutzerschnittstelle für die Nutzer gewährleistet werden.

Die detaillierte Definition der Anforderungen an Teilfunktionalitäten und an die Qualität und Sicherheit ist der Tabelle aus dem Anhang A.3 zu entnehmen.

### 3.2.3. Technische Ebene

Im Kapitel der Systemebene wurde logisch beschrieben, was der Prototyp können soll und über welche Schnittstellen diese Ziele erreicht werden sollen. Allerdings wurde noch nicht beschrieben, welche technischen Anforderungen sich für die Erreichung der Ziele ergeben.

**Problemstellungen** Damit das System nach dem vorgestellten Aufbau realisiert werden kann, muss das technische System bestimmte Probleme lösen:

ID	Problem	Quelle
<b>T-P-1</b>	Erzeugung eines cyber-physischen Systems als Messinstrument für die Anlagensimulation	
T-P-1.1	Lokale Verarbeitung der Messwerte	<i>S-FA-1</i>
T-P-1.1	Kommunikation zwischen Anlage und Geräteverwaltung	
<b>T-P-2</b>	Übergabe in die digitale Welt	<i>S-FA-2</i>
T-P-2.1	Einheitliche Kommunikation im Gesamtsystem	
T-P-2.2	Adressierbarkeit aller Komponenten	
T-P-2.3	Erzeugung eines digitalen Zwilling	
T-P-2.4	Datentransfer zum digitalen Zwilling	
T-P-2.5	Kategorisierung der Messwerte	
T-P-2.6	Generierung von Events	
<b>T-P-3</b>	Einbindung eines externen Benachrichtigungsdienstes	<i>S-FA-2</i>
<b>T-P-4</b>	Erzeugung einer Anwendung zur Visualisierung	<i>S-FA-3</i>
T-P-4.1	Hardware- und ortsunabhängige Verfügbarkeit der Informationen	
<b>T-P-5</b>	Sicherheit der Informationen	<i>S-QA-3</i>

Tabelle 6: Probleme aus technischer Ebene

**Technischer Aufbau des Systems** Für die Simulation einer Windenergieanlage wird ein kommunikationsfähiges System benötigt, welches Schnittstellen für Sensoren als Messmittel aufweist (T-P-1). Mit der Nutzung eines *Raspberry Pi 3 Model*

*B* können durch die lokale Nutzung von Python-Skripten die Messwerte verarbeitet werden. Anschließend können die Messwerte durch die Anwendung des REST-Paradigmas über ein Gateway an die SAP Cloud Platform gesendet werden. Die weiteren Probleme werden durch die Nutzung der SAP-Leonardo-Technologien gelöst. Diese werden im nachfolgenden Kapitel detaillierter analysiert. Daher wird der technische Aufbau im Detail in dem nachfolgenden Kapitel vorgestellt. Technisch soll das System dem Paradigma des Cloud-Computing, also der Nutzung von Microservices, folgen.

**Anforderungen** Die technische Umsetzung des Prototypen muss folgenden Anforderungen folgen:

ID	Anforderung	Quelle
<b>T-FA-1</b>	Die Simulation muss hardwareseitig rechnergestützt sein.	<i>T-P-1</i>
T-FA-1.1	Die Hardware muss Schnittstellen zur Sensoreinbindung aufweisen.	
T-FA-1.1	Die Software muss fähig sein, Berechnungen mit den empfangenen Sensorwerten zu durchzuführen.	
T-FA-1.1	Die Software muss fähig sein, Messwerte über eine REST-API an ein Gateway zu versenden.	
<b>T-FA-2</b>	Das Gateway muss fähig sein, die Messwerte zu empfangen.	
T-FA-2.1	Das Gateway muss fähig sein, die empfangenen Daten über eine REST-API zu versenden.	
T-FA-2.2	Das Gateway kann fähig sein, die erfassten Daten vor dem Versenden zu verarbeiten.	
T-FA-2.3	Das Gateway muss fähig sein, Befehle aus der SAP Cloud Platform zu empfangen.	
<b>T-FA-3</b>	Der Prototyp muss fähig sein, die Messwerte vom Gateway zu empfangen.	<i>T-P-2</i>
T-FA-3.1	Der Prototyp muss fähig sein, die Messwerte einer eindeutigen <i>DeviceId</i> zuzuordnen.	
T-FA-3.2	Der Prototyp muss fähig sein, der <i>DeviceId</i> eine eindeutige <i>GatewayId</i> zuzuordnen.	

ID	Anforderung	Quelle
T-FA-3.3	Der Prototyp muss fähig sein, die Messwerte einer <i>DeviceId</i> an einen digitalen Zwilling zu übergeben.	
T-FA-3.4	Der Prototyp muss fähig sein, die Metadaten einer <i>DeviceId</i> an einen digitalen Zwilling zu übergeben.	
T-FA-3.5	Der Prototyp muss fähig sein, Messwerte als <i>High</i> , <i>Medium</i> , <i>Low</i> zu kategorisieren.	
T-FA-3.6	Der Prototyp muss fähig sein, automatisch POST-Anfragen an die Web-API eines SNS zu senden.	
<b>T-R-1</b>	Der Prototyp muss die digitalen Zwillinge in einer UI5-Web-Applikation visualisieren.	
<b>T-R-2</b>	Der Prototyp und zugehörige Daten müssen in der SAP Cloud Platform gehostet sein.	

Tabelle 7: Anforderungen aus technischer Ebene

### **3.3. Systemanalyse und -entwurf**

Aufbauend auf die Anforderung, einen Prototypen mit den Technologien der SAP Leonardo IoT Foundation zu entwickeln, wird in diesem Kapitel die zugrundeliegende Systemarchitektur untersucht. Dabei erfüllt die Analyse mehrere Zwecke. Zum einen wird die Forschungsfrage FF-1.2 (s. 1.2), welche Möglichkeiten zur intelligenten Vernetzung die Architektur bietet, beantwortet. An die Beantwortung dieser Frage knüpft sich die Prüfung der Kompatibilität der Zielarchitektur mit der RAMI 4.0. In der Anforderungsanalyse wurde hauptsächlich die Anwendung des cloud-basierten Ansatzes als Lösung (s. 3.2.3) spezifiziert. Aus diesem Grund wird auf Grundlage der Analyse die angepasste Architektur des Zielsystems als Lösung für den technischen Aufbau entworfen.

#### **3.3.1. Systemarchitektur**

Wie in Abschnitt 2.3.2 beschrieben, bietet die SAP Leonardo IoT Foundation zahlreiche Dienste zur Integration von physischen Geräten in die SAP Cloud Platform und somit in die Geschäftswelt. Somit entstehen CPS, welche im Rahmen eines IoT-Projektes typischerweise drei Phasen durchlaufen: *Datentransport, Datenhaltung und Analyse* (s. Abschnitt 2.1.3). Wie die Architektur der SAP Leonardo IoT Foundation diesen Integrationsprozess bewerkstellt, ist in Abbildung 11 zu sehen. Die realen Geräte senden ihre Daten über *Gateways* mit verschiedenen Netzwerkprotokollen an die *SAP Cloud Platform Internet of Things Services* (1). Dort werden sie in einer PostgreSQL Datenbank gehalten (Acharya et al., 2019). Der Zugriff auf die Daten der CPS erfolgt im Rahmen einer service-orientierten Architektur (SOA) über verschiedene API. Um die Geräte in eine IoT-Anwendung zu integrieren, werden sie via *Message Processing* und das *IoT Application Enablement*<sup>2</sup> gesendet (2). Dort können digitale Zwillinge erstellt, funktionalisiert und analysiert werden. Für die Anwendungsentwicklung werden die Daten per OData- oder REST-Schnittstellen von der Cloud Foundry Umgebung an die WebIDE in der Neo Umgebung übergeben (3). Anschließend wird die Anwendung in die Cloud Foundry deployed. Die einzelnen Komponenten und Konzepte werden im Folgenden näher im Detail erläutert.

---

<sup>2</sup>heute: *SAP Leonardo IoT*

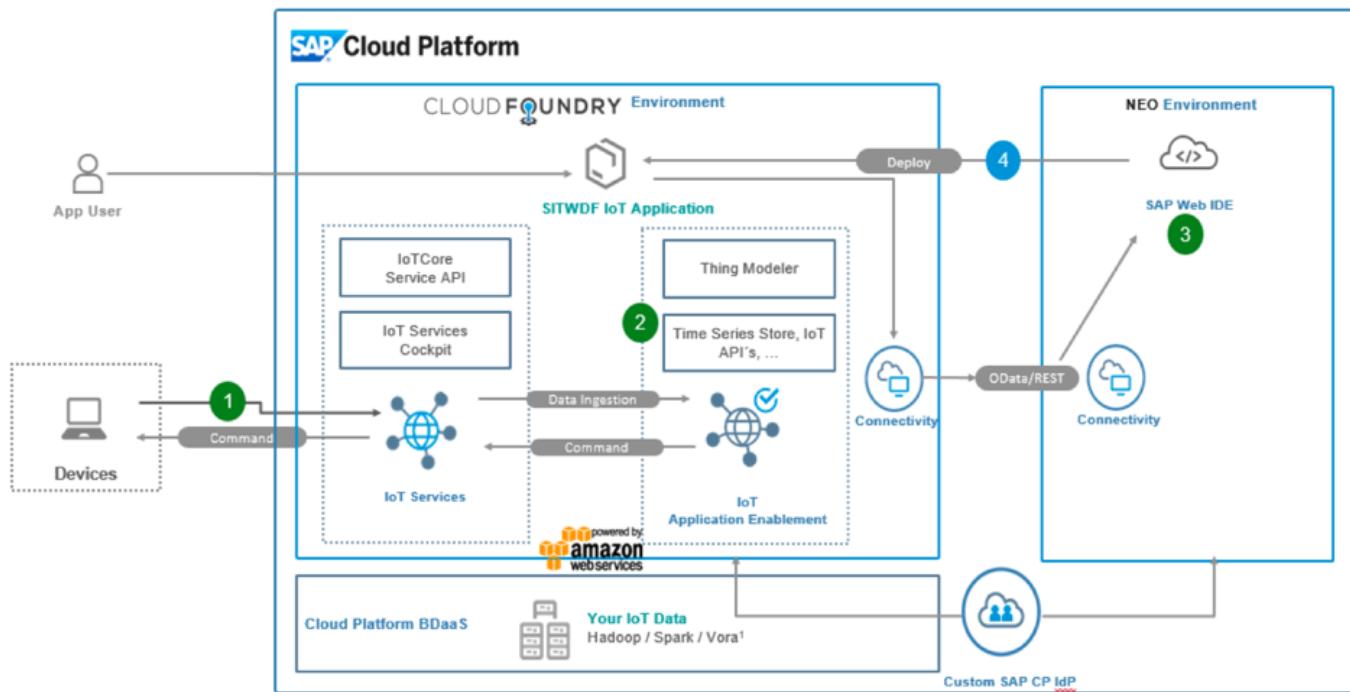


Abbildung 11: Architektur von SAP (Ganz, 2019)

### 3.3.2. Datentransport: Internet of Things Gateway

Der erste Schritt von einem realen physischen Ding zu einem CPS ist der Datentransport zu einem Datenobjekt im Netz (s. 2.1.3). Bezogen auf RAMI 4.0 und das Konzept der Industrie-4.0-Komponente (I-4.0-K) ist dies der Prozess der Integration und Kommunikation. Das reale Objekt wird an ihre Verwaltungsschale angebunden, um virtuell repräsentiert werden zu können. Die Architektur von SAP bietet für diesen Vorgang zwei Möglichkeiten. Entweder den direkten Transport der Daten in die Cloud über die *Gateway Cloud* oder über die *Internet of Things Edge Platform*. Neben dem Senden von Messwerten in die Cloud können außerdem aus der Cloud heraus Befehle an das Gerät gesendet werden. Welche Variante auszuwählen ist, hängt von individuellen Anwendungsfällen und benötigten Protokollen ab. Die *Gateway Cloud* wird in dem *SCP Internet of Things Service* standardmäßig für MQTT und REST mitgeliefert.

Die *Internet of Things Edge Platform* wird von den Entwicklern lokal auf dem Gerät nach dem ausgewählten Kommunikationsprotokoll (s. Tabelle 8<sup>3</sup>) selbst konfiguriert. Sie dient dazu, Messwerte von Geräten mit mangelnder Internetverbindung zu verarbeiten und bei verfügbarerer Verbindung an die Cloud zu senden. In beiden Fällen werden die Nachrichten mit den Messwerten im JavaScript Object Notation (JSON)- oder im Protocol Buffers (Protobuf)-Format mit POST-Anfragen an die API-Endpunkte der registrierten Geräte gesendet (SAP, 2020a). Das Registrieren der Geräte wird im nächsten Kapitel näher behandelt. SAP bietet außerdem die Möglichkeit, das *Internet of Things Service* mit dem *Internet of Things Edge Platform SDK* zu erweitern. Das SDK bietet Tools auf Eclipse, das Gateway mit *Interceptors* zu erweitern. Die Messwerte können vor dem Senden an die Cloud vom Gateway abgefangen, modifiziert oder gefiltert werden.

```

1      // Endpunkt der Gateway Cloud
2      https://<HOST_NAME>:443/iot/gateway/rest/measures/<
3          deviceAlternateId>
4      // Endpunkt des Edge Gateways
5      https://<IOT_GATEWAY_IP>:8699/measures/<deviceAlternateId>
```

Listing 1: API-Endpunkte der Gateways

### 3.3.3. Datenhaltung: SCP Internet of Things Service

Wie in Listing 1 zu sehen ist, haben die Gateways, welche die Daten der Geräte an die Cloud übermitteln, eine *deviceAlternateId* zum Ziel. Damit die Daten in der

Tabelle 8: Gateway-Protokolle

Protokoll	Cloud	Edge
MQTT	x	x
HTTP (REST)	x	x
CoAP		x
File		x
Modbus		x
OPC UA		x
SigFox		x
SNMP		x

<sup>3</sup>Quelle: SAP (2020a)

Cloud ankommen können, müssen die Geräte und Sensoren für die Messwerte im *Internet of Things Service* der SAP Cloud Platform registriert werden. Für diesen Zweck liefert SAP ein Datenmodell für die Geräte, nach dem eine Entität des Geräts erstellt werden muss (s. Abbildung 12). Nach dem Modell muss ein Gerät mindestens aus einem Sensor bestehen und eindeutig einem Gateway zugeordnet sein. Die Sensoren sind immer Instanzen von bestimmten Sensortypen, denen Fähigkeiten und Eigenschaften zugeordnet werden. Für die Registrierung bietet das System zwei Möglichkeiten. Man kann es entweder über die grafische Benutzeroberfläche des *IoT Services Cockpit* durch das Ausfüllen von Formularen erstellen oder über die *IoT Core Service API* (s. Abbildung 11). In beiden Fällen werden POST-Anfragen im JSON-Format an den API-Endpunkt des Tenants im *Internet of Things Service* gesendet. Für diesen Zweck stellt SAP eine Sammlung von *Device Management API* zur Verfügung. Nachdem die Geräte registriert wurden, kann die SAP Cloud Platform das Datenobjekt des CPS halten. Somit befindet sich das physische Ding in der Informationsschicht nach RAMI 4.0.

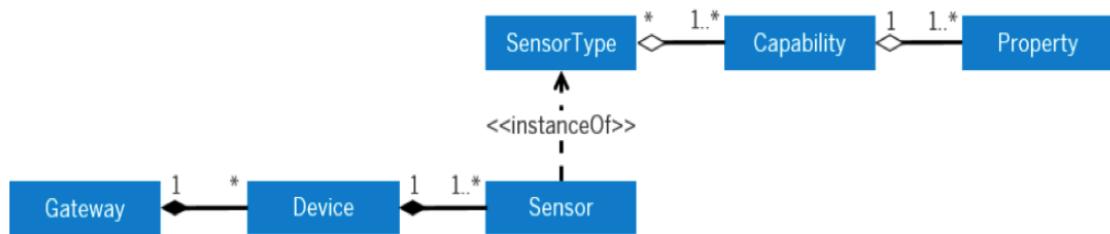


Abbildung 12: Gerätemodell

Genau so wie das Erstellen von Geräten über APIs, kann auf Messwerte und Eigenschaften der Geräte über GET-Anfragen an die API-Endpunkte zugriffen werden. Der Zugriff auf die Daten nennt sich in diesem Kontext *Message Processing* und kann über die von SAP zur Verfügung gestellte Sammlung von *Message Processing API* konfiguriert und abgerufen werden. Diese Schnittstellen sind vor allem hinsichtlich der Einbindung der Daten in externe Dienste und Anwendungen von großer Relevanz. Je nach Bedarf und Anwendungsfall können unterschiedliche Dienste konfiguriert werden (SAP, 2020a):

- *SAP Leonardo IoT Integration* für den Datentransfer an SAP Leonardo IoT

- *SQL* für den Fall, dass die Gerätedaten in einer externen DB gespeichert werden sollen
- *Kafka* z.B. für Big-Data Dienste
- *HTTP*, um einen eigenen API-Endpunkt zu nutzen

### **3.3.4. Analyse: SAP Leonardo IoT**

Über den Message Processing Service *SAP Leonardo IoT Integration* werden die Daten der Geräte an SAP Leonardo IoT übergeben. Laut SAP bietet Leonardo IoT „alle notwendigen Funktionen zum Einrichten von Thing- und Geschäftspartnerstrukturen als Repräsentation der realen Welt“ (SAP, 2019, S. 11). Mit allen notwendigen Funktionen sind in diesem Fall eine Sammlung von REST- und OData-basierten Microservices gemeint, die zum Speichern und Bereitstellen der Daten dienen. In erster Linie dienen sie dazu, einen digitalen Zwilling mit dem *Thing Modeler* des Geräts zu erstellen. Die folgende Abbildung (13) gibt eine Übersicht über die Softwarearchitektur und die Interaktion mit verschiedenen Komponenten. Mithilfe der API-Sammlung können zum Beispiel Geschäftspartner, Standorte, Berechtigungen oder Ereignisse zugeordnet werden. Dadurch wird dem *Ding*, welches vorher rohe Daten erzeugt hat, eine Funktion und ein Zweck zugeordnet. Nach der RAMI 4.0 befindet man sich somit in der Funktionsschicht. Eine weitere wichtige Funktion von SAP Leonardo IoT ist das *Application Enablement*, welches die modellierten Zwillinge an die Web IDE der Neo Laufzeitumgebung sendet. Dort können mit Hilfe von Templates *Freestyle IoT Applications* nach der UI5-Technologie von SAP erstellt werden. Außerdem dient SAP Leonardo IoT als Schnittstelle für weitere SaaS in unterschiedlichen Laufzeitumgebungen, für weitere Leonardo-Produkte mit Analysefunktionen oder für die Integration in Backend-Systeme. Allerdings handelt es sich bei SAP Leonardo um ein junges Projekt, welches die Funktionen zur Integration des Backend-Systems und die Einbindung der Analysefunktionen erst in Zukunft realisieren soll.

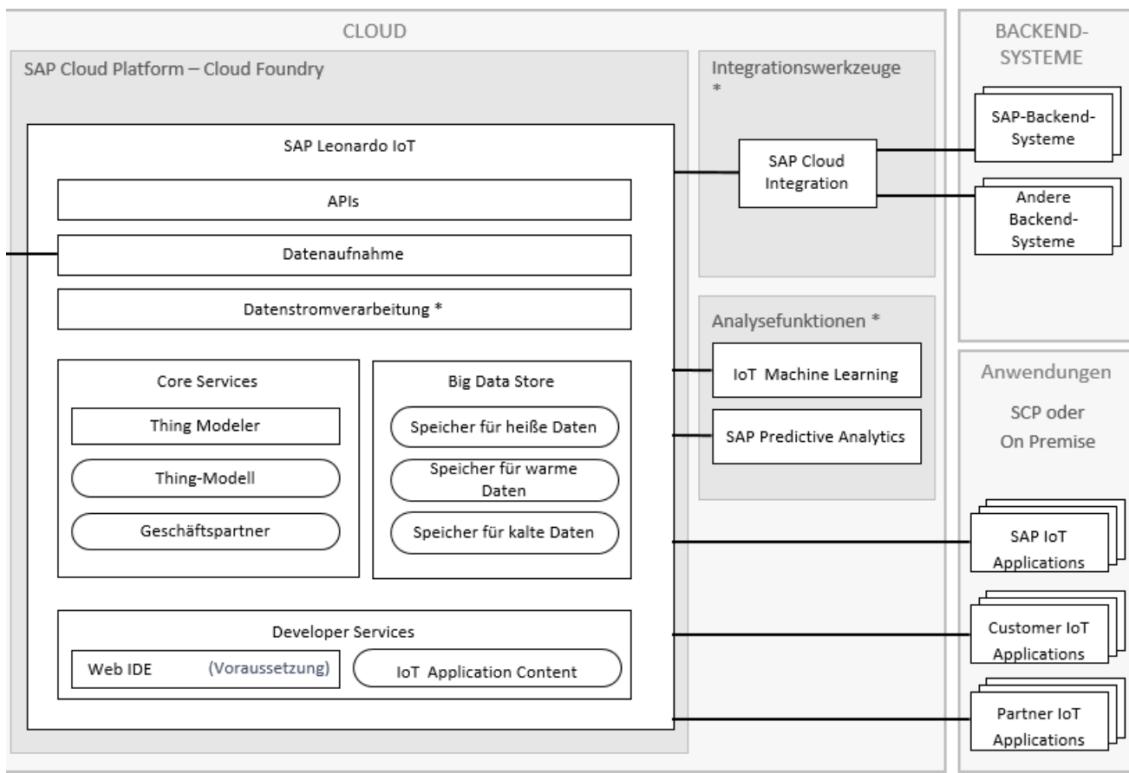


Abbildung 13: Architekturübersicht SAP Leonardo IoT (SAP, 2019, S. 12)

### 3.3.5. Destinations

Damit ein System mit Systemen aus verschiedenen Laufzeitumgebungen oder Endpunkten kommunizieren kann, werden in der SAP Cloud Platform Destinations erstellt. In der Abbildung 11 kann man z.B. erkennen, dass zwischen der Cloud Foundry Umgebung und der Neo Umgebung eine Verbindung hergestellt wird (3), um Thing-Daten für die Anwendungsentwicklung zu transferieren. Damit die Neo Umgebung die Daten empfangen kann, muss dort eine HTTP-Destination erstellt werden. Diese muss aus einer Ziel-URL des Cloud Foundry Dienstes und kann aus Authentifizierungsmethoden sowie Anmeldeinformationen bestehen.

### 3.3.6. Sicherheit

Wie in Abschnitt 2.2.2 beschrieben, trägt die Sicherheit in der Kommunikation in IoT-Netzwerken eine Schlüsselrolle. Insbesondere mit Blick auf die Architekturen-

delle der Systemlandschaft (s. Abbildung 11 und 13) wird deutlich, dass die Datenkommunikation im Rahmen einer SOA über zahlreiche Schnittstellen stattfindet. Der Zugriff auf diese Schnittstellen erfordert sichere Authentifizierungs- und Autorisierungsmaßnahmen. Die Infrastruktur von SAP stellt Standardmechanismen für die sichere Kommunikation zwischen den Komponenten zur Verfügung. Der Datentransport findet mit X509.1-Mechanismen asymmetrisch verschlüsselt über das *Transport Layer Security (TLS)* Protokoll statt (SAP, 2020b). Um die Geräte im Internet of Things Service registrieren zu können, erhält jeder Tenant einen client-spezifischen *private Key* im .pem- oder .p12-Format. Nur mit Angabe des Schlüssels können Edge Gateways so konfiguriert werden, dass über sie sicher mit dem Gerät kommuniziert werden kann. Auch der Zugriff auf die digitalen Zwillinge von SAP Leonardo IoT kann über das Berechtigungsmanagement eingeschränkt werden, sodass die Daten nach Verantwortlichkeiten getrennt werden. Der Zugriff auf die Ressourcen via API wird über die Autorisierung mit OAuth2.0 gesichert.

### 3.3.7. Kompatibilität mit Referenzarchitektur

Eine Anforderung an den Prototypen ist, Kompatibilität mit der RAMI 4.0 und dem Konzept der Industrie-4.0-Komponente aufzuweisen. Im Zuge der Systemanalyse wurden bereits für einzelne Komponenten Referenzen zu einzelnen Schichten der IT-Sicht des RAMI 4.0 erstellt. Abbildung 14 veranschaulicht die Referenzen zwischen den Schichten und den Komponenten des Systems.

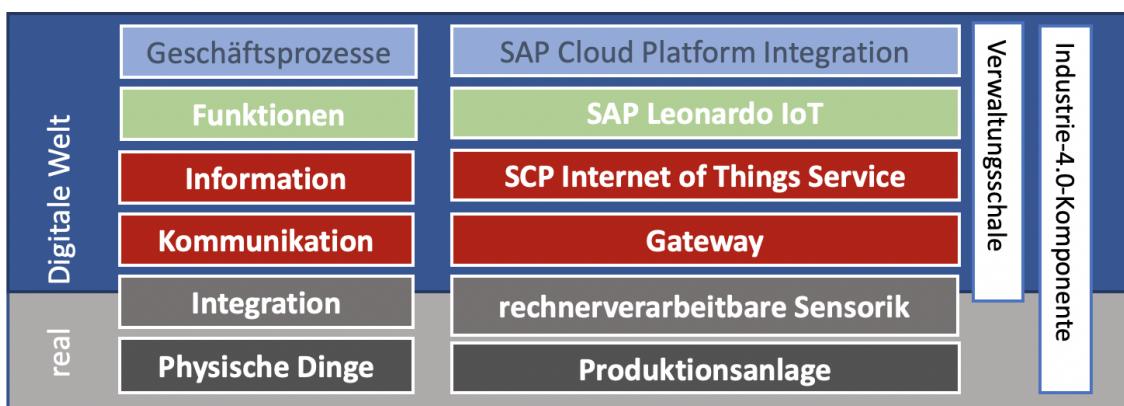


Abbildung 14: Referenz zu den Schichten der RAMI 4.0

Die Produktionsanlage wird an ihre Verwaltungsschale über das Gateway angebunden. Somit kann das Objekt als Industrie-4.0-Komponente bezeichnet werden. Über den Dienst SAP Cloud Platfrom Integration kann die horizontale Integration der Industrie-4.0-Komponente in das Backend und somit in die Organisation und Geschäftsprozesse erfolgen.

### **3.3.8. Systementwurf gemäß Architekturkonzept**

Nach dem vorliegenden Architekturkonzept und den Anforderungen an das System ergibt sich die in Abbildung 15 dargestellte Systemarchitektur. Sie spezifiziert sowohl die Komponenten des Systems als auch den Datenfluss. Für die Simulation einer Windenergieanlage wird ein Raspberry Pi über die IoT Edge Platform REST in die SAP Cloud Platform integriert. Die Daten des Geräts werden unter der deviceId der Entität in dem Internet of Things Service gehalten. Die grünen Pfeile visualisieren den Datenfluss von der Hardware über die SAP Cloud Plaform zu den Nutzern der IoT-Anwendung. Datenflüsse, die durch Anbindung von speziellen Microservices entstehen, werden durch die roten Pfeile repräsentiert. Dabei fließen die Daten durch die HTTP-Destinationen *SMS\_Gateway*, *Led\_Blink\_Command* und *Standard\_EventType*. Ausgelöst werden die Funktionen der Destinationen von Regeln, die für bestimmte Messwerte definiert werden. Zum Beispiel kann der Eingang von kritischen Werten das Senden von Befehlen zum Aufleuchten der LED oder das Senden einer Benachrichtigungs-SMS auslösen. Im folgenden Kapitel wird die Umsetzung detailliert beschrieben.

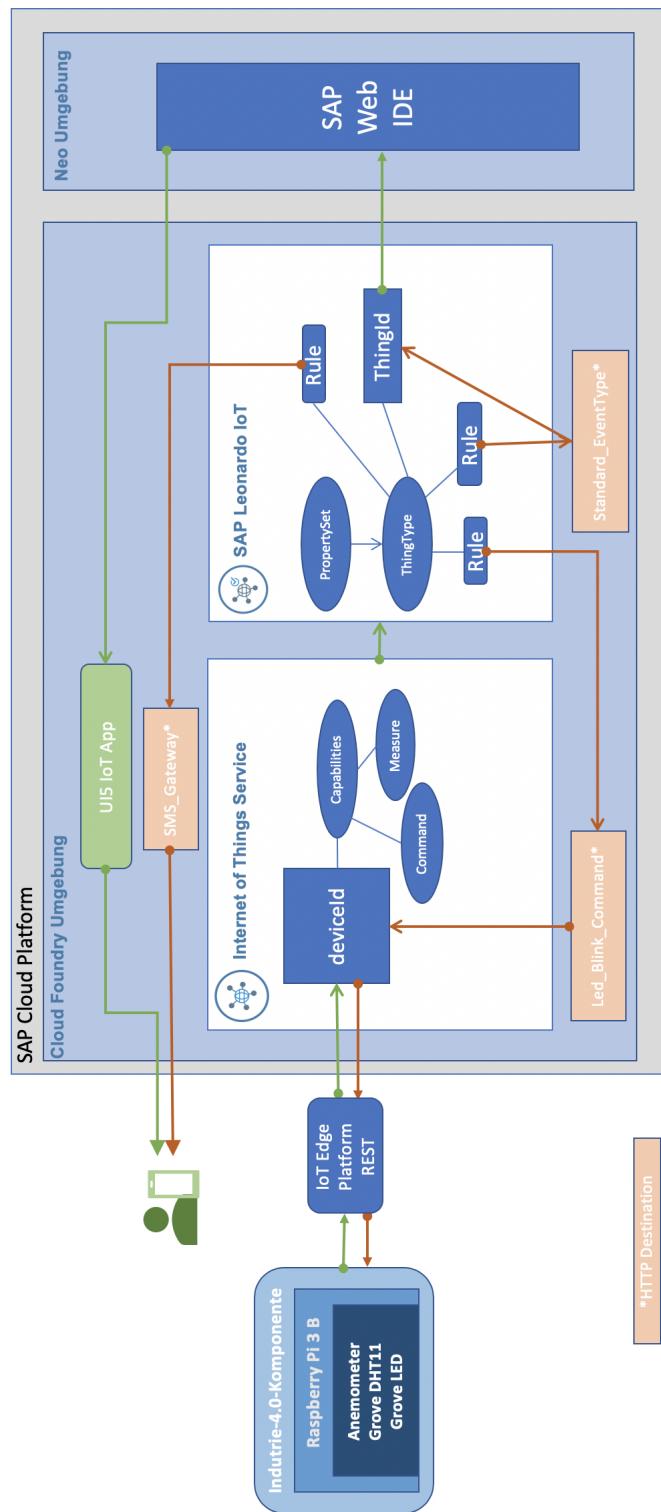


Abbildung 15: Systementwurf (eigene Darstellung)

## **3.4. Prototypische Implementierung des Anwendungsfalls**

In diesem Kapitel wird die Umsetzung des entwickelten Prototypen im Detail beschrieben. Zunächst wird die Erzeugung eines cyberphysischen Systems als Industrie-4.0-Komponente beschrieben. Im Anschluss wird dargestellt, wie Daten des Messinstruments über die *Internet of Things Edge Platform* an die SAP Cloud Platform übergeben werden. Daraufhin wird die Erzeugung eines digitalen Zwillings und die Zuordnung von Funktionen beschrieben. Abschließend wird dargestellt, wie der Zustand der Anlage den Nutzern in einer UI5-Applikation präsentiert wird.

### **3.4.1. Erzeugung eines cyberphysischen Systems**

Für die Simulation einer Windenergieanlage wird ein System erzeugt, welches Daten aus der Umwelt aufnehmen und verarbeiten kann. Bei den Daten wurde sich für die *Windgeschwindigkeit, Temperatur, Luftfeuchtigkeit, Luftdruck und Luftpumpe* entschieden. Als hardwareseitige Basis für die Simulation wird ein *Raspberry Pi 3 B* verwendet, welcher die erstellten *Python* Skripte zur Verarbeitung der Daten nutzt. Der Pi wird um die Zusatzplatine *Grove Pi + Board* erweitert. Somit wird der *DHT11-Sensor* zur Messung von Temperatur und Luftfeuchtigkeit über die 4-Pin-Schnittstelle *D4* einfach ohne Löten und Breadboard angebunden. Für die Messung der Windgeschwindigkeit ist die Entscheidung auf den Anemometer *Eltako Windsensor WS* gefallen. Im Gegensatz zum *DHT11-Sensor* ist der Anemometer ein aktives mechanisches Gerät, dessen Output Berechnungen erfordert, um die Windgeschwindigkeit zu erhalten. Bei den ausgegebenen Signalen handelt es sich um die Berührung zweier Metallkontakte, wobei zwei Signale eine Umdrehung ergeben. Ein Python Skript zählt die Signale über der GPIO-4-Schnittstelle und berechnet die Geschwindigkeit für ein Zeitintervall von 5 Sekunden. Aufgrund mangelhafter Sensorik wird der Luftdruck aus einem Array von realistischen Werten zufällig pro Zeitintervall ausgewählt. Gemeinsam mit der gemessenen Temperatur und dem Luftdruck wird die Luftpumpe berechnet. Bevor die Daten an das Gateway gesendet werden, werden sie dem String-Dictionary *sensorData* übergeben, um eine Anfrage im *JSON-Format* senden zu können. Außerdem wird eine rote Grove-LED über die digitale Schnittstelle D3 angebunden. Diese Komponente dient im Gegensatz zu den weiteren Komponenten dazu, bei kritischen Temperaturwerten Befehle zum Aufleuchten

zu erhalten. Damit soll gezeigt werden, dass sich der Kreis in einem cyberphysischen System von Sensorik bis Aktorik über SAP Leonardo als informationsverarbeitende Schicht schließt.

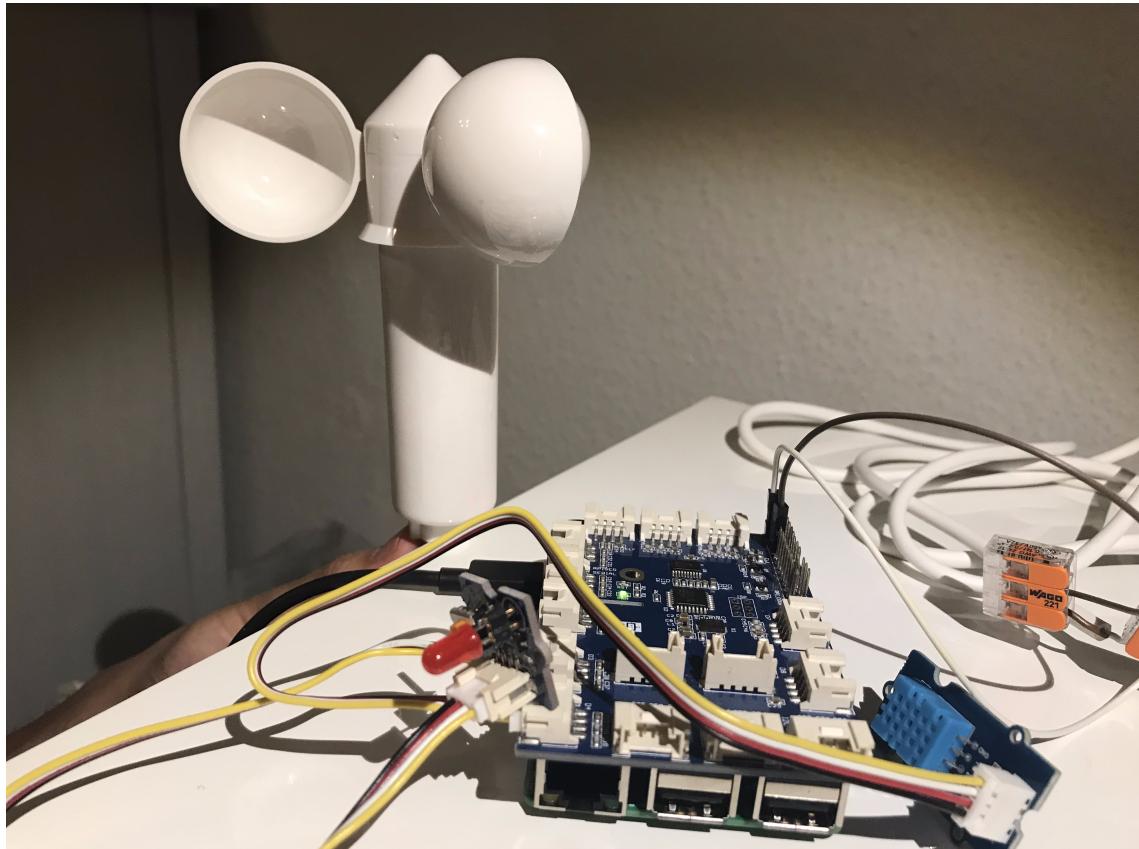


Abbildung 16: Cyberphysisches System für die Simulation (eigene Darstellung)

### 3.4.2. Einrichtung der Gateway Edge

Damit die aufgenommenen Sensorwerte zur Datenhaltung an den *Internet of Things Service* gesendet werden können, muss ein Gateway existieren. Doch anstatt die vordefinierten Cloud Gateways im *IoT Service Cockpit* zu nutzen, wird für den Prototypen ein lokales Edge Gateway des *REST*-Protokolls mit Hilfe der *Internet of Things Edge Platform* konfiguriert. Zunächst wird aus dem SAP Software Center eine Konfigurationsdatei heruntergeladen, welche auf dem Raspberry Pi lokal gespeichert wird. Mit dem Befehl `./build.sh REST` werden weitere Konfigurationsdateien

erzeugt, um die Verbindung zu dem Host des Internet of Things Service herzustellen. In der Datei *config\_gateway\_rest.xml* wird der Endpunkt für den Datentransfer in einer *ConnectionAddress* definiert und eine *GatewayAlternateId* definiert.

```
1 <cnf:address>https://5075f8b9-866e-4a4b-82f8-74687b72f1ab.eu10.cp.iot
  .sap:443/5075f8b9-866e-4a4b-82f8-74687b72f1ab/iot/core/api/v1/
  tenant/988439498</cnf:address>
2 <cnf:gateway gatewayAlternateId="1122334455667788">
```

Listing 2: Gateway-Verbindung zur Cloud

Als Serveradresse des Edge Gateways wird die IP-Adresse des Raspberry Pi angegeben. Somit ist die Zieladresse für die Sensordaten zum Beispiel:

```
1 https://192.168.178.52:8699/measures/<deviceAlternateId>
```

Listing 3: Zieladresse für Sensorwerte

Abschließend benötigt die Gateway-Registrierung einen *private Key im pem-Format*. Dafür wird eine GET-Anfrage an den Tenant des Internet of Things Service Hosts erstellt.

```
1 curl -X GET "https://5075f8b9-866e-4a4b-82f8-74687b72f1ab.eu10.cp.iot.
  sap/5075f8b9-866e-4a4b-82f8-74687b72f1ab/iot/core/api/v1/tenants
  /988439498/gatewayRegistrations/clientCertificate/pem"
```

Listing 4: GET-Anfrage für einen Client-Key

Nach dem Starten des Gateways auf dem Raspberry Pi mit dem Befehl *./gateway.sh* wird folgendes Gateway initialisiert:

```
1 {"id": "2019161729",
2 "alternateId": "1122334455667788",
3 "protocolId": "rest",
4 "name": "IoT Gateway REST",
5 "type": "edge",
6 "creationTimestamp": 1571741372988,
7 "status": "online",
8 "version": "4.42.0",
9 "operatingSystem": "Linux;4.19.75-v7+;arm"}
```

Listing 5: Gateway-Eigenschaften

### 3.4.3. Registrierung der Geräte

Bevor die Daten des Geräts von der SAP Cloud Platform gehalten werden können, muss eine Instanz eines Gerätetypen erstellt werden. Die Registrierung folgt dem vordefinierten Datenmodell aus Abbildung 12 und wird mit POST-Anfragen an die *Device Management API* durchgeführt. Zunächst wird die *Capability wind\_1* vom Typ *measure* erstellt. Der Capability werden die *Properties wind\_speed, temperature, humidity, pressure und airtight* zugeordnet. Damit das Gerät Befehle empfangen kann, wird außerdem die Capability *commands\_test* vom Typ *command* erstellt. Der Command hat eine *Property led vom Typ Boolean*. Anschließend werden beide Capabilities einem Sensortypen zugeordnet. Somit kann eine Instanz des Geräten erstellt werden, welchem das aktive Gateway (s. Listing 5) und ein Sensor vom Typ *test\_st\_com* hinzugefügt wird. Nachdem alle Entitäten hinzugefügt und zugeordnet wurden, hat das Gateway eine Zieladresse für den Datentransfer an die Cloud. Abbildung 17 bildet das Datenmodell der erstellten Geräteinstanz ab.

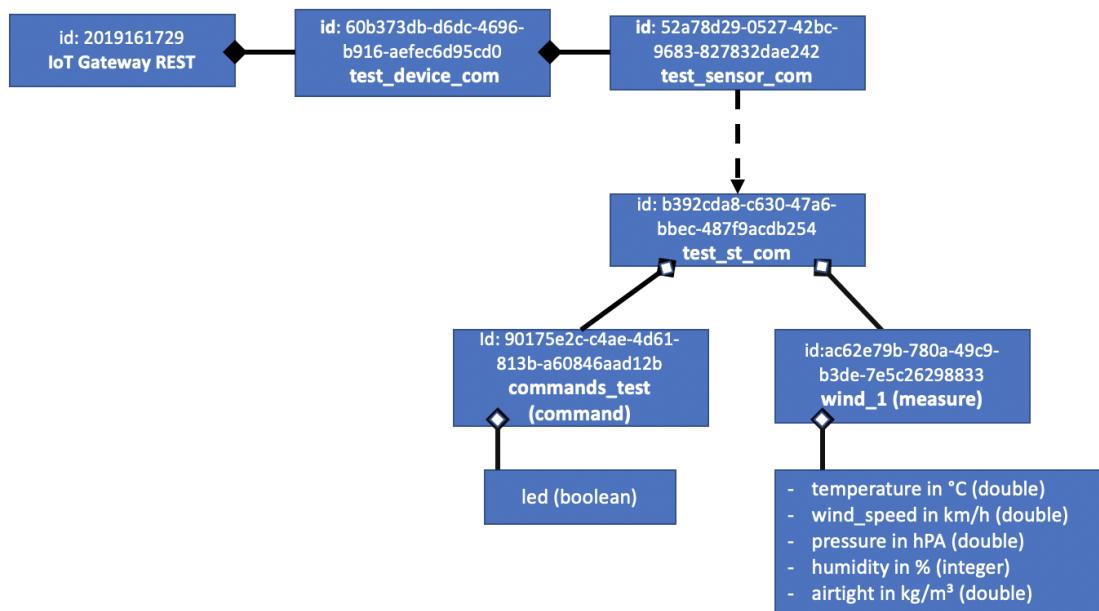


Abbildung 17: Gerätemodell der erstellten Instanz (eigene Darstellung)

### 3.4.4. Datentransfer an die Cloud

Mit der Registrierung des Gateways sowie des Geräts sind alle Bedingungen erfüllt, um die Messwerte an den Internet of Things Service zu senden. Nun wird aus dem Skript `test_data.py` zur Erfassung der Sensorwerte die Methode `send_data(sensorData)` des Skripts `send_test_data.py` aufgerufen. Somit kann ein *POST-Request* an das Edge Gateway gesendet werden. Die Anfrage besteht aus drei Teilen:

1. `postAdress: 'http://192.168.178.52:8699/measures/' + deviceAlternateId`
2. `data: JSON-Body aus capabilityAlternateId, sensorAlternateId, measures`
3. `header: 'content-type': 'application/json'`

Wie die gesendeten Daten im JSON-Body aussehen können, ist beispielhaft in Listing 6 dargestellt. Dass der Datentransfer erfolgreich ist, kann auf der Seite des Sendemediums mit der Antwort des Servers mit dem HTTP Code 202 bestätigt werden.

```
1  Reading sensor data ...
2  {'capabilityAlternateId': '1234', 'measures': [{'temperature':
    '19.0'}, {'wind_speed': '1.12102078977'}, {'pressure': '1010'}, {'humidity':
    '70.0'}, {'airtight': '1.2'}], 'sensorAlternateId':
    '1234'}
3  ==> HTTP Response: 202
```

Listing 6: Das Data-Feld der POST-Anfrage

Auf Empfängerseite kann der Empfang der Daten auf zweierlei Weisen bestätigt werden. Einerseits kann eine GET-Anfrage an die Zieladresse des Geräts im Internet of Things Service erstellt werden. Ergebnis der Anfrage sind die Messwerte im JSON-Format. Eine einfachere Möglichkeit bietet die Visualisierung der Daten im *IoT Service Cockpit*. Hier werden die empfangenen Daten sowohl im Diagramm als auch in einer Tabelle präsentiert (s. Abbildung 18). Mit Betrachtung der Zeitstempel für die empfangenen Daten lässt sich ebenfalls bestätigen, dass die Daten alle fünf Sekunden empfangen werden. Damit diese Daten in externe Dienste eingebunden

werden können, wird anschließend das *Message Processing* konfiguriert. Aus den verfügbaren Diensten wird entsprechend des Systementwurfs der Dienst „*Leonardo IOT*“ konfiguriert. Anschließend werden der Konfiguration Selektoren für die Messwerte und Metadaten der Geräte zugeordnet.

Date	Capability	Property	Value
23.01.2020, 13:34:45	wind_1	humidity	43
23.01.2020, 13:34:45	wind_1	temperature	35
23.01.2020, 13:34:45	wind_1	airtight	1.14
23.01.2020, 13:34:45	wind_1	pressure	1008
23.01.2020, 13:34:45	wind_1	wind_speed	0.693965250807
23.01.2020, 13:34:40	wind_1	airtight	1.15
23.01.2020, 13:34:40	wind_1	humidity	48
23.01.2020, 13:34:40	wind_1	pressure	1004
23.01.2020, 13:34:40	wind_1	temperature	32
23.01.2020, 13:34:40	wind_1	wind_speed	1.28116661688
23.01.2020, 13:34:35	wind_1	airtight	1.16
23.01.2020, 13:34:35	wind_1	humidity	49

Abbildung 18: Visualisierung der empfangenen Daten in Tabellenform

### 3.4.5. Erzeugung des digitalen Zwillings

Die Tatsache, dass der Zustand der Anlage im *IoT Service Cockpit* visualisiert wird, genügt für die Erfüllung der Anforderungen nicht. Der Grund dafür ist, dass die Daten lediglich für den Systemadministrator zugänglich sind. Um die Daten auch den Kunden bzw. dem Wartungspersonal zugänglich zu machen, wird eine UI5-Applikation erstellt. Diese bezieht die Daten aus der funktionalen Repräsentation

des Geräts, dem digitalen Zwilling. Mit der Konfiguration des *Message Processing* und den Selektoren können die Messwerte und Metadaten des Geräts an den digitalen Zwilling in *SAP Leonardo IoT* übergeben werden. Dafür wird zunächst ein neues *Thing Model* nach dem in Abbildung 17 dargestellten Gerätemodell erstellt. Genau so, wie im Internet of Things Service die Geräte, Capabilities und Konfigurationen für einen Tenant erstellt werden, findet die Modellierung in Leonardo IoT in *packages* statt. Die Modellierung wird vollständig über die in der grafischen Benutzerschnittstelle verfügbaren **Thing Engineering Funktionen** durchgeführt:

1. **Packages:** Erstellung des Pakets *windmills*
2. **Thing Properties Catalog:** Erstellung des *Property Sets wind\_1* analog zur Capability *wind\_1*
3. **Thing Modeler:** Modellierung des digitalen Zwillings
  - a) Erstellung des Thing-Typen *windenergieanlage* sowie Mapping zum Sensorotypen *test\_st\_com*
  - b) Erzeugung der Thing-Instanz *Enercon\_E126* sowie Mapping *test\_sensor\_com* (s. Abbildung 19)

Im Thing Modeler werden dem digitalen Zwilling außerdem Standortkoordinaten, ein Bild sowie die Beispielorganisation PowerSupply hinzugefügt. Mit den aktuellen Einstellungen besitzt der digitale Zwilling lediglich die Fähigkeit, die Messwerte ohne spezielle Funktionen abzubilden.

The screenshot shows the 'Verbindungsinformationen (5)' tab for a device named 'Enercon\_E126 (E126)'. A provider dropdown is set to 'SAP Cloud Platform IoT Service for Cloud Foundry Environment'. The 'Sensor-/Mapping-Auswahl' section shows a mapping named 'map\_com\_meas' for a sensor named 'test\_st\_com'. The 'Mapping zwischen Thing-Modell und Gerätemodell' table lists five attributes: airtight, humidity, pressure, temperature, and wind\_speed, each mapped to their respective capabilities in the Thing model.

Property Sets oder ...	Zugeor...	Geräteattribut	Capability	Sensortyp	
wind_1					
airtight	←	airtight	ac62e79b-780a-49c...	b392cda8-c630-47a...	
humidity	←	humidity	ac62e79b-780a-49c...	b392cda8-c630-47a...	
pressure	←	pressure	ac62e79b-780a-49c...	b392cda8-c630-47a...	
temperature	←	temperature	ac62e79b-780a-49c...	b392cda8-c630-47a...	
wind_speed	←	wind_speed	ac62e79b-780a-49c...	b392cda8-c630-47a...	

Abbildung 19: Mapping zwischen Thing und Sensor

### 3.4.6. Visualisierung in einer UI5-Anwendung

Das Erstellen eines Thing Models und einer Thing-Instanz genügt zunächst, um den Anlagenzustand in einer Webapplikation zu visualisieren. Für die Visualisierung steht in der Web IDE eine Leonardo-IoT-Erweiterung mit Vorlagen zur Erstellung von IoT-Anwendungen zur Verfügung. Vor der Nutzung der Vorlagen muss der Zugriff auf die modellierten Zwillinge und deren Daten gewährleistet werden. Weil die Web IDE sich in der Neo Umgebung der SCP und die digitalen Zwillinge sich in der Cloud Foundry Umgebung befinden, werden Vertrauensbeziehungen zwischen den Umgebungen hergestellt. Nachdem in der Neo Umgebung die Destinationen der Leonardo IoT Funktionen angelegt wurden, können die Daten übergeben werden. Im Wizard zum Erstellen der Anwendung wird als Datenquelle der Service

IOTAS-ADVANCEDLIST-THING-ODATA und das Property Set cf.devbeta.-windmills:wind\_1 gewählt. Anschließend wird die Erstellung folgender Seiten konfiguriert:

- Startseite mit einer Landkarte zur Verortung der Anlagen und Liste der Anlagen (s. Abbildung 20)
- Eine Karte mit einer Übersicht zum Anlagenzustand (s. nächstes Kapitel)
- Eine Seite mit Details der Anlage (s. Abbildung 21)
- Analyseseite für die Visualisierung der Messwerte (s. Abbildung 22)

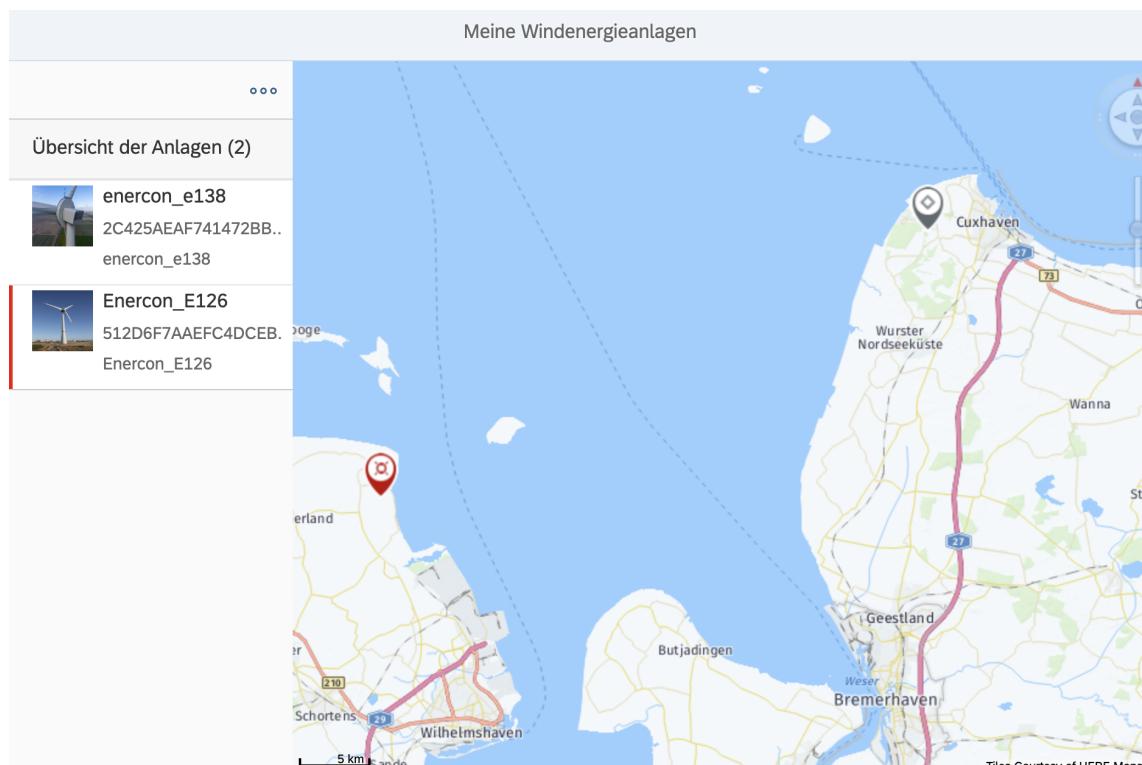


Abbildung 20: Startseite der Anwendung

The screenshot shows a web-based application interface for managing IoT assets. At the top, there's a header with a back arrow and the text "Details zur Anlage". Below the header, there's a small icon of a wind turbine and the text "Enercon\_E126 Enercon\_E126". Underneath this, there are four tabs: "ALLGEMEINE DATEN", "MESSWERTE" (which is currently selected), "EREIGNISSE", and "ZEITSTRAHL".

The main content area is titled "MESSWERTE". It contains a search bar with the placeholder "Suchen" and icons for search, refresh, and settings. Below the search bar is a table with the following columns: "Measured Value Name", "Current Value", "Min Error", "Min Warning", "Max Error", "Max Warning", and "Last Measured".

Measured Value Name	Current Value	Min Error	Min Warning	Max Error	Max Warning	Last Measured
wind_1.airtight	1.19					13:33 vor 1 Tag
wind_1.humidity	43					13:34 vor 1 Tag
wind_1.pressure	1013.0					13:34 vor 1 Tag
wind_1.temperature	27.0		20.0		25.0	13:34 vor 1 Tag
wind_1.wind_speed	1.2811667		0.5		80.0	13:34 vor 1 Tag

Abbildung 21: Details zur Anlage

Neben den allgemeinen Daten und den Messwerten zu der Anlage bietet die Seite eine Übersicht über die Ereignisse, die für bestimmte Messwerte generiert werden. Nach der Erstellung der Anwendung in der Web IDE können in SAP Leonardo IoT weiterhin neue Funktionen erstellt werden, die zur Laufzeit in die Anwendung integriert werden. Im nächsten Kapitel wird die Erzeugung von Ereignissen näher behandelt.

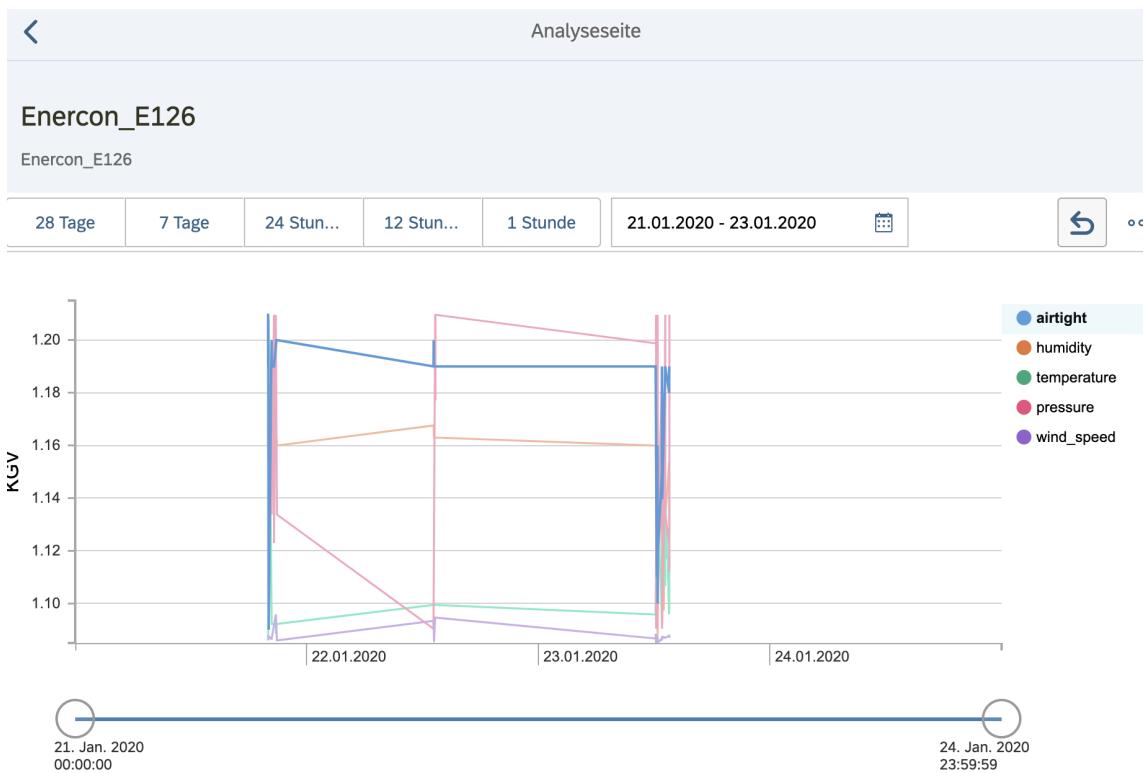


Abbildung 22: Analyseseite

### 3.4.7. Regeln, Ereignisse und Aktionen

Damit aus den eingehenden Messwerten Informationen gewonnen werden kann, werden im nächsten Schritt Regeln für die Daten und darauffolgende Aktionen definiert. Für dieses Vorhaben stellt SAP Leonardo IoT Funktionen über die Benutzeroberfläche bereit. Ziel ist es, bei Eingang von *kritischen Temperaturwerten*, dem Wartungspersonal eine *Benachrichtigungs-SMS* zu senden, dem Raspberry Pi einen *Befehl zum Aufleuchten der LED* zu senden und ein *Ereignis* für den Zwilling in der Anwendung anzuzeigen. Außerdem sollen Ereignisse generiert werden, wenn die *Windgeschwindigkeit* die Eigenschaften *stiller Wind, Sturm oder Orkan* aufweisen.

**Regelkontext** Als Bezugspunkt für die Regeln wird der Regelkontext `windea_context_` erstellt. Hier wird lediglich definiert, dass sich die Regeln auf das Property Set `wind_1` des Thing-Typen `windenergieanlage` beziehen sollen.

**Regeln** Im Regeleditor werden anschließend die expliziten Regeln für eingehende Messwerte definiert. Als Regelkontext wird für jede Regel *windea\_context\_* angegeben. Für diesen Anwendungsfall werden vier Regeln nach dem Schema in Abbildung 23 definiert:

- *windea\_regel\_temp*: Wenn die Temperatur größer ist als 25, wird ein Event *HighTemp* generiert
- *wind\_sturm*: Wenn die Windgeschwindigkeit 6 übersteigt, wird ein Event *Sturm*
- *wind\_windstille*: Wenn die Windgeschwindigkeit kleiner ist als 1.85, wird ein Event *Windstille* generiert
- *wind\_orkan*: Wenn die Windgeschwindigkeit 8 übersteigt, wird ein Event *Orkan* generiert

The screenshot shows two panels: 'Regelkontext' (Rule Context) and 'Regeleditor' (Rule Editor).

**Regelkontext:**

- Regelkontext\*:
- Datenobjekt:

**Regeleditor:**

- If
- Then
  - IoT\_Event\_Name:
  - IoT\_Event\_Severity:

Abbildung 23: Definition der Regel *windea\_regel\_temp*

**Aktionen** Die oben definierten Regeln dienen zunächst nur der Konditionsüberprüfung und lösen bisher keine Aktionen aus. Mit Aktionen sind in diesem Fall HTTP-POST-Anfragen an bestimmte Server oder Endpunkte gemeint, welche der Cloud Foundry Umgebung als Destination hinzugefügt werden. In den Fällen der Ereigniserzeugung und dem Befehl zum Blinken der LED sind bereits vordefinierter Schnittstellen vorhanden. Für die Auslösung der Benachrichtigungs-SMS wird jedoch eine eigene *API mit Python* erstellt. Im Folgenden werden die einzelnen Schritte detaillierter erläutert.

**Aktion: AWS\_Notation** Wie in dem Datenflussdiagramm (Abbildung 9) dargestellt, interagiert ein externer Benachrichtigungsdienst mit dem System in der SAP Cloud Platform. Als Nachrichtendienst wurde der *Simple Notification Service* von Amazon Web Services gewählt. AWS bietet ein Software Development Kit für Python, welches sich *Boto 3* nennt. Mit Nutzung der `boto3`- sowie der `Flask`-Bibliothek wurde eine API für den Client der AWS entwickelt. Die API hat unter der Route `/post.json` eine POST-Methode für eine Telefonnummer und eine da-zugehörige Nachricht:

```
1 client = boto3.client("sns",
2     aws_access_key_id="AKIAI44Y3WVEWIZYG4DA",
3     aws_secret_access_key="EXKQJR2D*****",
4     region_name="eu-west-1")
5
6 logger.info('sending sms ...')
7 client.publish(
8     PhoneNumber="{!!}.format(phone),
9     Message="{!!}.format(message)
10 )
```

Listing 7: Flask-API für AWS SNS

Anschließend wurde eine *manifest.yml*-Datei erstellt, um der der App für das Deployen in die Cloud Founry Umgebung die notwendigen Ressourcen zuzuweisen. Daraufhin wurde die Applikation einfach über das Command Line Interface (CLI) von Cloud Foundry in die SAP Cloud Platform deployed:

```

1 cf api https://api.cf.eu10.hana.ondemand.com
2 cf login
3 cf push AWS_SMS

```

Listing 8: Deployment in die SAP Cloud Platform

Schließlich wurde diese App mit der zugewiesenen URL (s. Listing 9) als Destination in die Cloud Foundry Umgebung integriert. Um die Auslösung der SMS zu implementieren, wurde zum Schluss in SAP Leonardo IoT die Aktion *AWS\_Notification* definiert. Die Aktion wird durch die Regel *windexe\_regel\_temp* ausgelöst und sendet die in Listing 9 dargestellte POST-Anfrage.

```

1 POST to Destinattion AWS_Notification: https://awssms-agile-lizard.
  cfapps.eu10.hana.ondemand.com/postjsonals
2
3   Payload:
4   {
5     "phone": "004915772661219" ,
6     "message": "Temperatur uebersteigt 25 Grad. Massnahmen einleiten"
7 }

```

Listing 9: JSON Payload an Destination

**Aktion: Led\_Blink** Das Senden des Befehls zum Aufleuchten der roten LED am Raspberry Pi folgt der selben Regel wie die Aktion zum Auslösen der SMS. Der Unterschied besteht in dem Endpunkt für die Anfrage. Der Befehl zum wird an die Zieladresse des Geräts für *commands* im Internet of Things Service gesendet. In der Anfrage-Payload werden die explizit der Sensor und die Capability des Geräts adressiert.

```

1 POST to Destination Led_Blink_Command: https://5075f8b9-866e-4a4b-82f8-
  -74687b72f1ab.eu10.cp.iot.sap/5075f8b9-866e-4a4b-82f8-74687b72f1ab/
  iot/core/api/v1/tenant/988439498/devices/60b373db-d6dc-4696-b916-
  aefec6d95cd0/commands
2 Payload:
3   {

```

```

4     "sensorId": "52a78d29-0527-42bc-9683-827832dae242",
5     "capabilityId": "90175e2c-c4ae-4d61-813b-a60846aad12b",
6     "command": {
7         "led": true
8     }
9 }
```

Listing 10: POST-Anfrage zum Senden den Befehls

Als Folge der Anfrage wird unter der Capability comamnds\_t est der Wert der Property led=true gesetzt. Da die Anweisung bisher nur an die virtuelle Instanz des Geräts gesendet wurde, muss der Raspberry Pi eine GET-Anfrage an das Gateway des Geräts im IoT Service senden. Um die Befehle so schnell wie möglich zu erhalten, wird dem Skript zum Senden der Messwerte zur Laufzeit die Methode get\_commands hinzugefügt. Wenn der Antwort-String den Werte true enthält, wird ein Befehl zum Aufleuchten an die Schnittstelle des Sensors gesendet.

```

1 r = requests.get('http://192.168.178.52:8699/commands/60b373db-d6dc
-4696-b916-aefec6d95cd0)
```

Listing 11: GET-Anfrage für Commands an das Gateway

**Aktion: Events generieren** Alle Aktionen zur Eventgenerierung haben die gleiche API als Zieldestination. Für jede Regel wird jeweils eine eigene Aktion generiert, welche eine POST-Anfrage an die API des von SAP vordefinierten Ereignistypen StandardEventType auslöst. Es unterscheiden sich lediglich die Werte im Payload je nach auslösender Regel. Variiert wird zwischen der EventSeverity (1 = High, 2 = Medium, 3 = Low), der EventInfo und der ExternalId. Im folgenden Listing ist beispielhaft am Fall der Regel *wind\_windstille* eine Anfrage dargestellt. Die weiteren Anfragen sind im Detail im Anhang gelistet.

```

1 POST to Destination Standard_EventType: https://events-sap.cfapps.
eu10.hana.ondemand.com/ES/EventType/com.sap.appiot.eventtypes:
StandardEventType/v1/Events
2 {
3     "BusinessTimestamp": "${event.time}",
```

```

4   "Type": "Alert",
5   "EventInfo": "Alert on Wind_Speed",
6   "EventStatus": "Open",
7   "EventSeverity": 3,
8   "EventCode": null,
9   "EventSource": null,
10  "ThingId": "${thing.id}",
11  "ThingProperty": "wind_speed",
12  "ExternalId": "Windstille"
13 }

```

Listing 12: Beispieldatenanfrage für die Generierung eines Ereignisses

In der UI5-Anwendung machen sich die generierten Ereignisse auf zweierlei Weisen sichtbar. Erstens kann auf der Startseite eine Karte zur Übersicht über den Anlagenzustand geöffnet werden (s. Abbildung 24). Jeweils nach `EventSeverity` werden die Ereignisse als *Error*, *Warning* oder *Information* eingestuft.

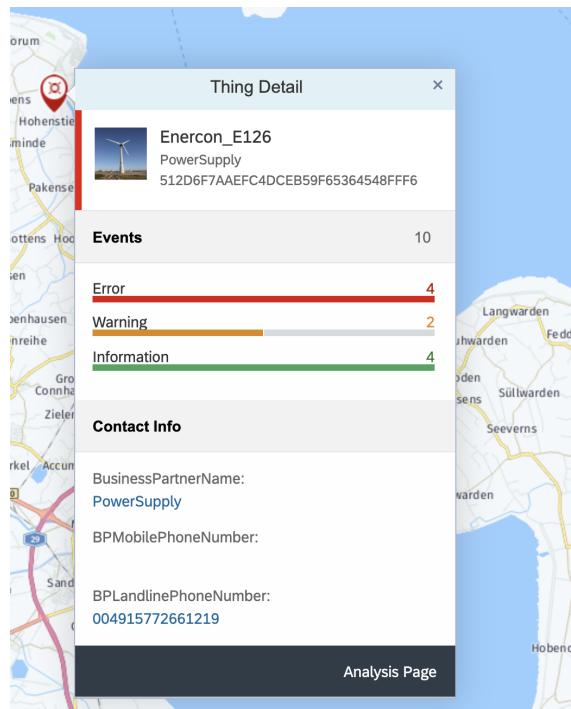
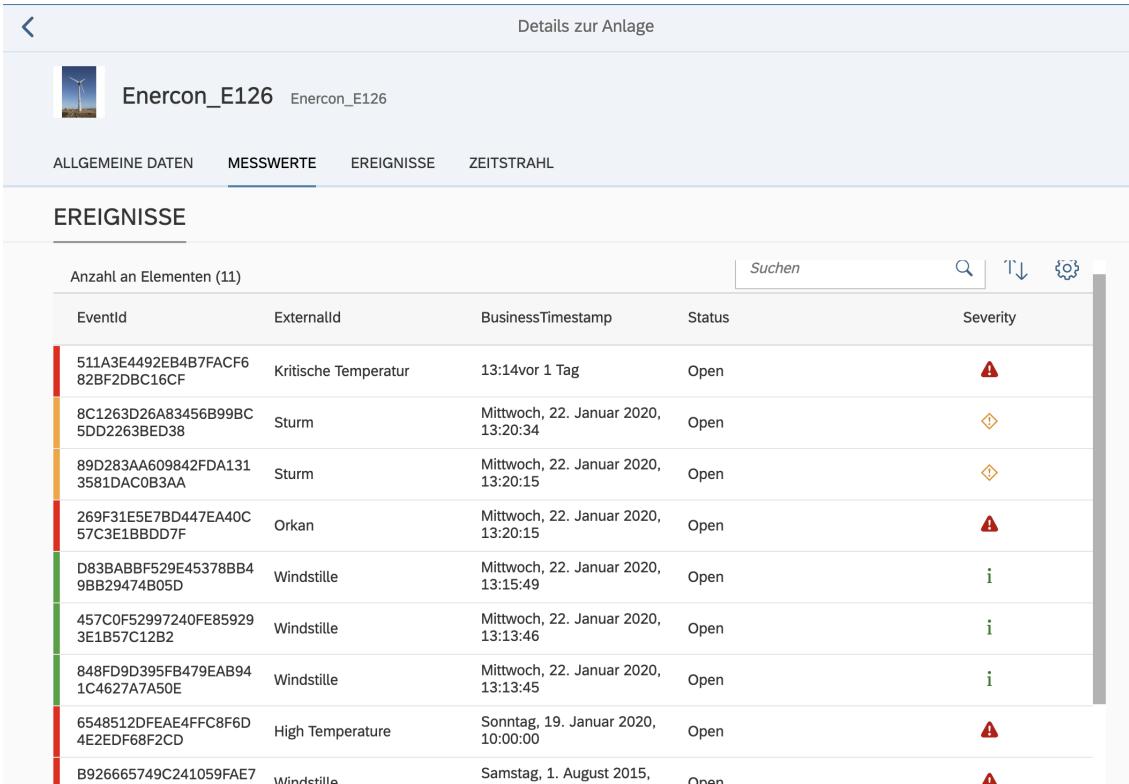


Abbildung 24: Übersicht über den Zustand der Anlage

Die in der Übersicht grob klassifizierten Ereignisse werden in der Detailansicht der Anlage mit den dazugehörigen Werten und Informationen aufgelistet (s. Abbildung 25).



EventId	ExternalId	BusinessTimestamp	Status	Severity
511A3E4492EB4B7FACF682BF2DBC16CF	Kritische Temperatur	13:14vor 1 Tag	Open	⚠
8C1263D26A83456B99BC5DD2263BED38	Sturm	Mittwoch, 22. Januar 2020, 13:20:34	Open	⚠
89D283A609842FDA1313581DAC0B3AA	Sturm	Mittwoch, 22. Januar 2020, 13:20:15	Open	⚠
269F31E5E7BD447EA40C57C3E1BBDD7F	Orkan	Mittwoch, 22. Januar 2020, 13:20:15	Open	⚠
D83BABB529E45378BB49BB29474B05D	Windstille	Mittwoch, 22. Januar 2020, 13:15:49	Open	ℹ
457C0F52997240FE859293E1B57C12B2	Windstille	Mittwoch, 22. Januar 2020, 13:13:46	Open	ℹ
848FD9D395FB479EAB941C4627A7A50E	Windstille	Mittwoch, 22. Januar 2020, 13:13:45	Open	ℹ
6548512DFEAE4FFC8F6D4E2EDF68F2CD	High Temperature	Sonntag, 19. Januar 2020, 10:00:00	Open	⚠
B926665749C241059FAE7D9116305E59	Windstille	Samstag, 1. August 2015, 21:24:00	Open	⚠

Abbildung 25: Detaillierte Auflistung der Ereignisse

### 3.4.8. Zusammenfassung der Implementierung

Die Implementierung in diesem Kapitel beschreibt die Umsetzung des in Abschnitt 3.3.8 entworfenen Gesamtsystems. Als Industrie-4.0-Komponente wurde ein Raspberry Pi mit Messinstrumenten zur Erfassung von Umweltdaten ausgestattet. Durch die Konfiguration eines Edge Gateways auf Basis des REST-Protokolls wurde das physische Medium in der SAP Cloud Platform als Cyber-physisches System abgebildet. Anschließend wurde die virtuelle Repräsentanz ganz nach dem Paradigma der cloud-nativen Entwicklung (s. Abschnitt 2.2.4) durch die Nutzung von Microservices mit Funktionen ausgestattet. Als Ergebnis der Datenintegration und

Virtualisierung steht eine SAP UI5 GUI zur Verfügung, welche die Möglichkeit bietet, den Zustand einer Anlage einzusehen und zu bewerten.

## 4. Evaluation

In diesem Kapitel wird untersucht, ob und wie der umgesetzte Prototyp den Anforderungen an das System gerecht wird. Grundlage der Untersuchung ist das Testen des Prototypen. Ähnlich wie die Anforderungsanalyse, erfolgt auch die Evaluation für die Abstraktionsebenen Kontextebene, Systemebene und technische Ebene. Auf Basis dieser Evaluationen wird schließlich eine passende Handlungsempfehlung für den Anwendungsfall gegeben.

### 4.1. Evaluation auf Kontextebene

In der Kontextebene wurden Anforderungen an das System gestellt, welche sich aus den Einflussfaktoren im Systemkontext ergeben. Besonders die Probleme im Zusammenhang mit der Branche der Energiewirtschaft bilden die Kernanforderungen an das System (s. Anhang A.1). Die sich daraus ergebenden funktionalen Anforderungen (K-FA-1) können im Großen und Ganzen als erfüllt betrachtet werden. Angefordert war es, dem Nutzer die Einsicht auf den digitalen Zwilling einer Anlage zu gewähren, einschließlich der Anzeige von Messwerten, Standorten und prädiktiven Informationen. Mit Betrachtung der Abbildungen 21, 20, 24 und 25 scheinen die Anforderungen erfüllt: Die Anlage wird auf der Startseite auf einer Landkarte verortet, die Übersicht über den Zustand liefert Bewertungen des Zustands und die Detailansicht listet einzelne Messwerte auf. Anhand der Einbindung des SNS von AWS, aber auch des Befehls zum Aufleuchten der LED, kann dem Wartungspersonal die Reaktion auf kritische Zustände ermöglicht werden. Probleme entstehen bei der Betrachtung der Anforderung *Echtzeit*. Wie in Abbildung 18 dargestellt, werden die Daten sofort an die Cloud transferiert. Allerdings benötigt die Reaktion auf die Daten, also die Aktivierung der HTTP-Anfragen durch die Aktionen, eine Zeitspanne zwischen 20 Sekunden und 2 Minuten. Diese Zeitspanne ist zwar immer noch kürzer als der 10-Minuten-Takt des SCADA-Systems, erfüllt aber nicht die Anforderung *Echtzeit*. Nichtsdestotrotz werden die definierten Anwendungsfälle durch den Proto-

typen realisiert. Auch die qualitativen Anforderungen können größtenteils als erfüllt betrachtet werden. Die SOA von SAP Leonardo und der SAP Cloud Platform liefert die Flexibilität, das System an Veränderungen anzupassen, neue Funktionen einzubinden und neue Geräte hinzuzufügen. In diesem Prototypen wurde dies anhand von Destinationen realisiert. Zudem können über Schnittstellen weitere intelligente Dienste von SAP Leonardo und andere Funktionen (s. Abbildung 13) angebunden werden. Dass die Randbedingungen (K-RA-2 und K-RA-3) zur Orientierung an der RAMI 4.0 erfüllt werden, wird nach der Systemanalyse deutlich. Die Architektur von SAP Leonardo wird auf die Schichten der IT-Sicht von RAMI 4.0 angewandt. Das Ergebnis wird in Abbildung 14 dargestellt. Allerdings wird in der Umsetzung der Kommunikation im Prototypen nicht, wie empfohlen, der OPC-UA-Standard verwendet. In der Systemanalyse wurde lediglich erwähnt, dass die Nutzung der OPC-UA über die IoT Edge Platform möglich ist.

## 4.2. Evaluation auf Systemebene

Wie das System die Anforderungen aus der Kontextebene erfüllen soll, wurde in der Systemebene definiert. Der innerere logische Aufbau des umgesetzten Prototypen erfüllt mit einigen Schwachstellen die Anforderungen aus Anhang A.3. Das Messinstrument sollte alle 5 Sekunden Daten erfassen und gemäß den Eigenschaften einer Industrie-4.0-Komponente an ein übergeordnetes IT-System senden (S-FA-1). Mit Be trachtung der Abbildung 18 wird die Erfüllung dieser Anforderungen bestätigt. Auch die Virtualisierung der Anlage (S-FA-2) in einem digitalen Zwilling einschließlich der Funktionalisierung der eingehenden Daten wurde in der Implementierung erfolgreich umgesetzt. Schwachstellen zeigen sich eher aus qualitativer Sicht als aus funktionaler Sicht. Beim Testen des Prototypen unter Erhöhung der Temperatur über den angegebenen Grenzwert, wurde eine Unzuverlässigkeit des Systems beobachtet. Während bei einigen Versuchen die Aktionen wie das Senden der SMS und das Aufleuchten der LED sofort aktiviert wurden, gab es neben Versuchen mit Verzögerungen auch einige Fehlschläge. Der Empfang der SMS wird in Abbildung 26 bestätigt. Die Ursache der Schwachstellen konnte allerdings nicht identifiziert werden. Es werden lediglich Verzögerungen im *Message Processing* vermutet. Dies zeigt sich auch darin, dass die im Internet of Things Service zuverlässig empfangenen Messwerte sich

nicht sofort in dem digitalen Zwilling der IoT-Anwendung aufzeigen. Ausgenommen von diesen Einzelheiten ist die Qualität des Systems gewährleistet. Ein Kriterium bezüglich der Eigenschaften einer Industrie-4.0-Komponente ist die Kommunikation in einem einheitlichen semantischen Datenformat. Sichergestellt wurde die einheitliche Kommunikation mit der Nutzung des JSON-Datenformats in jeglichen APIs. Zudem wurde die Kommunikation über das Gateway durch das Einfügen von *private Keys* für den Tenant der IoT Services gesichert. Außerdem durchlaufen die API-Anfragen Authentifizierungsmethoden mit Hilfe von OAuth 2.0. Somit kann auf die Vertraulichkeit der Informationen gezählt werden.

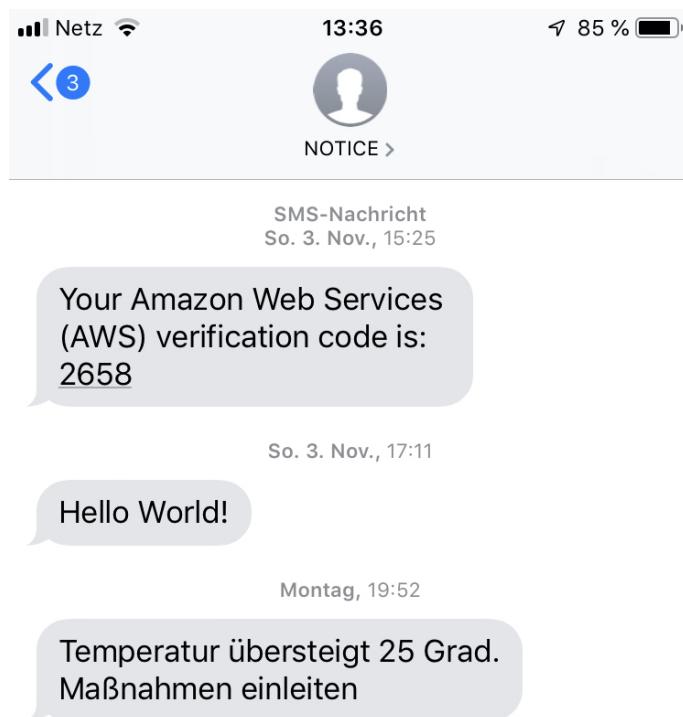


Abbildung 26: Empfang der Benachrichtigungs-SMS

### 4.3. Evaluation auf technischer Ebene

Dass die technischen Anforderungen (s. Kapitel 3.2.3) an den Prototypen größtenteils erfüllt sind, kann damit bestätigt werden, dass der Prototyp auf Systemebene nahezu alle erforderlichen Eigenschaften aufweist. Die funktionalen Anforderungen

lassen sich grob in drei Teile aufteilen. Der erste Teil betrifft die Erzeugung eines CPS (T-FA-1). Mit der Nutzung eines mit dem Internet verbundenen Raspberry Pi und der Anbindung entsprechender Sensorik wurde ein kommunikationsfähiges und rechnergestütztes System entworfen. Als nächstes wurden Anforderungen an das Gateway definiert (T-FA-2). Durch die lokale Konfiguration einer Gateway Edge auf Basis des REST-Protokolls wurde eine Industrie-4.0-Komponente erzeugt, indem das physische Medium an ihre Verwaltungsschale angebunden wurde. Das zuverlässige Senden der Messwerte über das Gateway kann bestätigt werden. Auch die Verzögerungen im Empfang der Befehle aus SAP Leonardo IoT sind nicht mit der Unzuverlässigkeit des Gateways, sondern der des Message Processing zu begründen. Als letztes wurden funktionale Anforderungen für die Virtualisierung des CPS definiert (T-FA-3). Auch diese Anforderungen erfüllt der Prototyp. Die Messwerte werden zuverlässig vom einem eindeutigen Gateway empfangen, einer *deviceId* zugeordnet und über das Message Processing an den digitalen Zwilling übergeben. Dort werden für die Messwerte, durch die Definition von Regeln und Aktionen, Ereignisse mit den Schweren *High*, *Medium*, *Low* (s. Listing 12) generiert. Zudem werden automatische POST-Anfragen an die Web-API des AWS SNS gesendet. Das Verhalten der SNS-App wird über ein Kibana Dashboard beobachtet.

#### 4.4. Handlungsempfehlung

Anbindung der realen Anlagen über das OPC-UA Protokoll mit der Gateway Edge Anbindung an HANA Datenbank, da schnellere In-Memory Verarbeitung als Post-GreSQL Forschungsfragen Anforderungen Warten bis der Scheiß ausgereift ist aber auf jeden Fall an RAMI orientieren

Komplett Anforderungsanalyse abgleichen Unflexible UI-Entwicklung weil Module und Packages nicht dokumentiert OPC UA in Edge Möglich aber im Prototypen nicht implementiert

Abbildung 18 kann entnommen werden, dass die Daten zuverlässig im 5-Sekunden-Takt empfangen werden. Problematisch ist die Verarbeitung in der Leo-Umgebung. API Reaktionszeit schlecht und fehleranfällig und unzuverlässige Datenverarbeitung im SAP Leonardo IoT Regelwerk, Regeln werden manchmal einfach nicht ausgelöst.

Das System bietet sicherlich mehr Möglichkeiten und ist bestimmt stabiler. Trotz

umfassender API-Dokumentation ist der Nutzer des Systems nicht ausreichend mit Anwenderdokumentation versorgt. Hilfreich sind die Blog-Beiträge der SAP-Community, welche jedoch nur anwendungsfallsbasierte Informationen liefern.

## **5. Kritische Würdigung und Ausblick**

Was war das Ziel der Arbeit? Forschungsfragen integrieren Wie kann die digitale Transformation in der Energiewirtschaft durchgeführt werden

Vieles wird noch nicht unterstützt Reflexion: Was hab ich gemacht? (Selbst-Kritisches) z.B. scheiß-Edge und SAP sehr BETA und schlecht dokumentiert blabla  
Ausblick: Weitere Möglichkeiten Integration mit SAP Backend HANA DB APIs/SDK für Leonardo Edge Processing mit Interceptors and Adapters, echtes Gerät mit echten Sensorwerten statt RPI und teilweise simulierte Werte

Beantwortung der Frage: wie gut man mit SAP Leonardo digital transformieren kann auch nach RAMI 4.0

## Literatur

- Acharya, G., Bajaj, G., Dhar, A., Ghosh, A., und Lahiri, A. (2019). *SAP Cloud Platform: Cloud-Native Development*. Rheinwerk Verlag GmbH.
- Adolphs, P. (2017). Smart Products / Innovative Produktentwicklung. In Schulz, T., Herausgeber, *Industrie 4.0: Potenziale umsetzen und erkennen*. Vogel Business Media.
- AWS (2019). Informationen zu AWS. Abgerufen 23.12.2019, von <https://aws.amazon.com/de/about-aws/>.
- Barthelmä, N., Flad, D., Haußmann, T., Kupke, T., Schneider, S., und Selbach, K. (2017). Industrie 4.0 – Eine industrielle Revolution? In *Industrie 4.0*, Seiten 33–56. Springer Fachmedien Wiesbaden.
- Bauer, W., Schlund, S., Marrenbach, D., und Ganschar, O. (2014). Industrie 4.0- Volkswirtschaftliches Potenzial für Deutschland.
- Bauernhansl, T. (2014). *Die Vierte Industrielle Revolution - Der Weg in ein wertschaffendes Produktionsparadigma*, Seiten 5–35. Springer Fachmedien Wiesbaden, Wiesbaden.
- Bendel, O. (2019a). Cyber-physische Systeme. Abgerufen 20. November 2019, von <https://wirtschaftslexikon.gabler.de/definition/cyber-physische-systeme-54077/version-369944>.
- Bendel, O. (2019b). Industrie 4.0. Abgerufen am 19. November 2019, von <https://wirtschaftslexikon.gabler.de/definition/industrie-40-54032/version-368841>.
- Beuth publishing DIN (2016). DIN SPEC 91345:2016-04. Beuth Verlag Berlin. Abgerufen 2. November, von <https://www.beuth.de/de/technische-regel/din-spec-91345/250940128>.
- Bitkom e.V (2015). Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0. Abgerufen 19. Novemver 2019,

von <https://www.bitkom.org/sites/default/files/file/import/150410-Umsetzungsstrategie-0.pdf>.

Bundesministerium für Wirtschaft und Energie (2015). Die Energie der Zukunft Vierter Monitoring Bericht zur Energiewende. Abgerufen 16.12.2019, von <https://www.bmwi.de/Redaktion/DE/Publikationen/Energie/vierter-monitoring-bericht-energie-der-zukunft.pdf>.

Bundesministerium für Wirtschaft und Energie (2019). Digitale Transformation in der Industrie. Abgerufen 19. November 2019, von <https://www.bmwi.de/Redaktion/DE/Dossier/industrie-40.html>.

Doleski, O. D. (2016a). Handlungsdruck – Rahmenbedingungen beschleunigen die Veränderungen in der Energiewirtschaft. In *Utility 4.0*, Seiten 5–10. Springer Fachmedien Wiesbaden.

Doleski, O. D. (2016b). *Utility 4.0: Transformation vom Versorgungs- zum digitalen Energiedienstleistungsunternehmen*. Springer Fachmedien Wiesbaden.

Doleski, O. D. (2017). *Herausforderung Utility 4.0: Wie sich die Energiewirtschaft im Zeitalter der Digitalisierung verändert*. Springer Fachmedien Wiesbaden. Hrsg.

Drath, R. (2016). Technische Grundlagen. In Heinze, R., Herausgeber, *Industrie 4.0 im internationalen Kontext : Kernkonzepte, Ergebnisse, Trends*. VDE Verlag GmbH, Berlin Offenbach.

Dzombeta, S., Kalender, A., und Schmidt, S. (2017). Datensicherheit bei Smart Services und Cloud-Sicherheit und Datenschutz im Cloud-Computing. In Schulz, T., Herausgeber, *Industrie 4.0: Potenziale umsetzen und erkennen*. Vogel Business Media.

Elsner, M., González, G., und Raben, M. (2018). *SAP Leonardo: Konzepte, Technologien, Best Practices*. Rheinwerk Verlag GmbH.

Fraunhofer-Gesellschaft (2016). Trends für Industrie 4.0. Abgerufen 19. November 2019, von <https://www.fraunhofer.de/content/dam/zv/de/Forschungsfelder/Produktion-Dienstleistung/Trends-fuer-Industrie-40.pdf>.

- Fraunhofer ISE (2019). Energiewende – Paradigmenwechsel und Digitalisierung. Abgerufen 15. November 2019, von <https://www.ise.fraunhofer.de/de/leitthemen/energiewende-digital.html>. Fraunhofer ISE.
- Ganz, B. (2019). SAP Cloud Platform Solution Diagrams & Icons. Abgerufen 12. Novemver 2019, von <https://d.dam.sap.com/a/JPUXye>.
- Gartner (2017). Prognose zur Anzahl der vernetzten Geräte im Internet der Dinge (IoT) weltweit in den Jahren 2016 bis 2020 (in Millionen Einheiten) [Graph]. In *Statista*. Abgerufen 22. November 2019, von <https://de.statista.com/statistik/daten/studie/537093/umfrage/anzahl-der-vernetzten-geraete-im-internet-der-dinge-iot-weltweit/>.
- Hänisch, T. (2017). Grundlagen Industrie 4.0. In *Industrie 4.0*, Seiten 9–31. Springer Fachmedien Wiesbaden.
- Howells, R. (2017). Welcome to the Digital Renaissance. Abgerufen 18. November, von <https://news.sap.com/2017/07/welcome-to-the-digital-renaissance/>.
- Hänisch, T. (2017). Grundlagen Industrie 4.0. In *Industrie 4.0*, Seiten 9–31. Springer Fachmedien Wiesbaden.
- Hübner, I. (2017). Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0). In *Industrie 4.0 im internationalen Kontext : Kernkonzepte, Ergebnisse, Trends*. VDE Verlag GmbH, Berlin Offenbach.
- Hübschle, K. (2017). Digitale Anwendungen / Datenverarbeitung in der Industrie. In Schulz, T., Herausgeber, *Industrie 4.0: Potenziale umsetzen und erkennen*. Vogel Business Media.
- IREB e.V. (2017). IREB Certified Professional for Requirements Engineering - Foundation Level -. Abgerufen 25.12.2019, von <https://www.ireb.org/de/downloads/>.
- Kenn, H. (2016). Architekturen für das Internet der Dinge. In Heinze, R., Herausgeber, *Industrie 4.0 im internationalen Kontext : Kernkonzepte, Ergebnisse, Trends*. VDE Verlag GmbH, Berlin Offenbach.

- Krone, O. und Bachmann, M. (2017). Digitalisierung im Verteilnetz: Evolution oder Revolution anhand konkreter Beispiele. In *Herausforderung Utility 4.0*, Seiten 451–464. Springer Fachmedien Wiesbaden.
- Köster, M. und Mache, T. (2017). Digital Transformation Canvas – Übersicht behalten und Handlungsfelder gestalten. In *Herausforderung Utility 4.0*, Seiten 355–382. Springer Fachmedien Wiesbaden.
- Lauenroth, K., Schreiber, F., und Schreiber, F. (2016). *Maschinen-und Anlagenbau im digitalen Zeitalter: Requirements Engineering als systematische Gestaltungskompetenz für die Fertigungsindustrie Industrie 4.0*. Beuth Verlag.
- Lea, P. (2018). *Internet of Things for Architects*. Packt Publishing, Birmingham.
- Lüth, C. (2016). Funktion und Herausforderungen von Cyber-Physical Systems. In Heinze, R., Herausgeber, *Industrie 4.0 im internationalen Kontext : Kernkonzepte, Ergebnisse, Trends*. VDE Verlag GmbH, Berlin Offenbach.
- mind square (2019). Amazon Web Services (AWS). Abgerufen 23.12.2019, von <https://mindsquare.de/knowhow/amazon-web-services/>.
- Radtke, M. und Litzel, N. (2019). Was ist Big Data? Abgerufen 22. November 2019, von <https://www.bigdata-insider.de/was-ist-big-data-a-562440/>.
- Roth, A. (2016). Industrie 4.0 - Hype oder Revolution? In *Einführung und Umsetzung von Industrie 4.0*, Seiten 1–15. Springer Berlin Heidelberg.
- SAP (2019). Services von SAP Leonardo IoT. Abgerufen 23.11.2019, von <https://help.sap.com/doc/a48fdbd924724b378d6f71c54c9f35b5/1903a/de-DE/leoAPI-de.pdf>.
- SAP (2020a). Internet of Things Gateway. Abgerufen 18.01.2020, von <https://help.sap.com/doc/0820e88affdf445ab9b2eba586ce49fc/Cloud/en-US/iot.cf.gateway.cloud.and.iot.edge.platform.pdf>.
- SAP (2020b). Security - SAP Cloud Platform Internet of Things. Abgerufen 17.01.2019, von

<https://help.sap.com/doc/b4cc900aa36d46e390ea7bd77ab1f89b/Cloud/en-US/iot.cf.security.pdf>.

Sendler, U. (2016). Die Grundlagen. In *Industrie 4.0 grenzenlos*, Seiten 17–39. Springer Berlin Heidelberg.

Umweltbundesamt (2018). Was ist ein Smart-Grid? Abgerufen 16.12.2019, von <https://www.umweltbundesamt.de/service/uba-fragen/was-ist-ein-smart-grid>.

Utecht, M. und Zierau, T. (2018). *SAP fürr Energieversorger: Neue Perspektiven für die Zukunft der Versorgungswirtschaft*. SAP Press. Rheinwerk Verlag GmbH.

Voigt, K.-I. (2018). Industrielle Revolution. Abgerufen 20. Novemver, von <https://wirtschaftslexikon.gabler.de/definition/industrielle-revolution-38116/version-334867>.

## A. Anforderungen

### A.1. Anforderungen aus Kontextebene

ID	Anforderung	Quelle
<b>K-FA-1</b>	Das System muss dem Nutzer Zugriff auf den digitalen Zwilling der Anlage gewähren.	<i>K-P-1</i>
K-FA-1.1	Das System muss dem Nutzer die aktuellen Messewerte in Echtzeit anzeigen.	<i>K-P-1.1</i>
K-FA-1.2	Das System muss dem Nutzer die Verortung der Anlage ermöglichen.	
K-FA-1.3	Das System muss dem Nutzer prädiktive Informationen liefern.	
K-FA-1.4	Das System muss dem Nutzer die Reaktion auf kritische Zustände in Echtzeit ermöglichen.	<i>K-P-1.2</i>
<b>K-QA-1</b>	Die Architektur des Systems muss dem SA die flexible Anpassung an Änderungen erlauben.	<i>K-P-2</i>
<b>K-QA-2</b>	Die Architektur des Systems muss dem SA die Einbindung neuer Anlagen erlauben.	<i>Auftraggeber</i>
<b>K-QA-3</b>	Die Architektur des Systems muss dem SA die Einbindung von intelligenten Diensten erlauben.	<i>K-P-4.1</i>
<b>K-QA-4</b>	Die Architektur des Systems muss dem SA erlauben, das System um Funktionen zu erweitern.	<i>K-P-4.1</i>
<b>K-RA-1</b>	Für die Umsetzung des Prototypen muss die SAP Leonardo IoT Foundation verwendet werden.	<i>K-P-4</i>
<b>K-RA-2</b>	Die Architektur des Systems muss mit RAMI 4.0 konform sein.	<i>K-P-3</i>
<b>K-RA-3</b>	Die Simulation muss die Eigenschaften einer Industrie-4.0-Komponente aufweisen.	<i>K-P-4.2</i>

Tabelle 10: Anforderungen aus Kontextebene

## A.2. Lösung aus Kontextebene

ID	Lösung	Quelle
<b>K-L-1</b>	Intelligente Verwaltungsschale für die reale Anlage	<i>K-P-1</i>
K-L-1.1	Der Zustand und zugehörige Daten sollen jederzeit einsehbar sein	<i>K-P-1.1</i>
K-L-1.2	Der Zustand der Anlage soll bewertbar sein	<i>K-P-1.2</i>
K-L-1.3	kritische Zustandsveränderungen sollen unverzüglich gemeldet werden	<i>K-P-1.3</i>
<b>K-L-2</b>	IT-Sicht des RAMI 4.0 und Industrie-4.0-Komponente	<i>K-P-3</i>
<b>K-L-3</b>	Prototypische Architekturvorlage für IoT-Projekte	<i>K-P-3</i>
K-L-3.1	Messinstrument zur Simulation einer realen Anlage	<i>K-P-4.2</i>

Tabelle 12: Lösungen aus Kontextebene

## A.3. Anforderungen aus Systemebene

ID	Anforderung	Quelle
<b>S-FA-1</b>	Das Messinstrument muss min. alle 5 Sekunden Umgebungsdaten erfassen und verarbeiten können.	<i>Kontext</i>
S-FA-1.1	Das Messinstrument muss die Messungen nach der Erfassung kommunizieren können.	<i>I-4.0-K</i>
S-FA-1.2	Das Messinstrument muss als Entität in einem übergeordneten IT-System vorliegen.	<i>I-4.0-K</i>
<b>S-FA-2</b>	Der Prototyp muss das Messinstrument virtuell als Anlage beschreiben können.	<i>I-4.0-K</i>
S-FA-2.1	Der Prototyp muss das Messinstrument eindeutig per <i>URI</i> identifizieren können.	<i>I-4.0-K</i>
S-FA-2.2	Der Prototyp muss das Messinstrument eindeutig per Koordinaten verorten können.	<i>I-4.0-K</i>

ID	Anforderung	Quelle
S-FA-2.3	Der Prototyp muss die Daten (Messwerte und Eigenschaften) der virtuellen Repräsentation halten.	<i>I-4.0-K</i>
S-FA-2.4	Der Prototyp muss die Grenzüberschreitung der empfangenen Daten erkennen.	<i>Kontext</i>
S-FA-2.5	Der Prototyp muss die empfangenen Daten kategorisieren.	<i>Kontext</i>
S-FA-2.6	Der Prototyp muss eine Benachrichtigungs-SMS versenden.	<i>Kontext</i>
S-FA-2.7	Der Prototyp muss Ereignisse generieren.	<i>Kontext</i>
S-FA-2.8	Der Prototyp muss die Anlage an eine Anwendung übergeben.	<i>Kontext</i>
<b>S-FA-3</b>	Der Prototyp muss dem Nutzer eine Benutzerschnittstelle zur Verfügung stellen.	<i>Kontext</i>
S-FA-3.1	Die Benutzerschnittstelle muss alle Anlagen auf einer Karte verorten.	
S-FA-3.2	Die Benutzerschnittstelle muss die schnelle Bewertung des Zustands ermöglichen.	
S-FA-3.3	Die Benutzerschnittstelle muss alle Messwerte auflisten.	
S-FA-3.4	Die Benutzerschnittstelle muss die Messwerte visualisieren.	
<b>S-QA-1</b>	Die Benutzerschnittstelle muss intuitiv sein.	
<b>S-QA-2</b>	Die Architektur des Systems muss dem SA die horizontale Integration der Anlage ermöglichen.	<i>RAMI 4.0</i>
<b>S-QA-3</b>	Die Kommunikation muss nach einem einheitlichen semantischen Modell erfolgen.	<i>I-4.0-K</i>
<b>S-QA-3</b>	Der Prototyp muss sicher sein.	<i>I-4.0-K</i>
S-QA-3.1	Der Prototyp muss die Verfügbarkeit der Informationen gewährleisten.	<i>I-4.0-K</i>
S-QA-3.2	Der Prototyp muss die Vertraulichkeit der Informationen gewährleisten.	<i>I-4.0-K</i>
S-QA-3.3	Der Prototyp muss die Integrität der Informationen gewährleisten.	<i>I-4.0-K</i>

---

ID	Anforderung	Quelle
----	-------------	--------

---

Tabelle 13: Anforderungen aus Systemebene

## B. Quelltext

### B.1. Python Skript: test\_data.py

```

1
2 # Skript zum Erzeugen und Lesen von Sensordaten
3
4 import grovepi
5 from grovepi import *
6 import math
7 import random
8 import sys
9 from gpiozero import DigitalInputDevice
10 from time import sleep
11 from send_test_data import send_data
12 import send_test_data as send
13 import json
14
15 # Wind/Anemometer
16 count = 0
17 radius_cm = 2.0          # Radius of the anemometer
18 interval = 5             # How often to report speed
19 ADJUSTMENT = 1.18        # Adjustment for weight of cups
20 CM_IN_A_KM = 100000.0
21 CM_IN_METER = 1000.0
22 SECS_IN_AN_HOUR = 3600
23
24 # Luftdichte
25 airtight = 0
26
27 #Temperatur und Luftfeuchtigkeit
28
29 temp_hum_sensor = 4    # digital port 4.
30

```

```

31 blue = 0      # The Blue colored sensor.
32 white = 1     # The White colored sensor.
33
34 # Collection fuer Luftdruckwerte
35 pressureCollection = [1002, 1004, 1005, 1006, 1008, 1010, 1011, 1012,
   1013]
36
37 def simulatePressure():
38     global pressureCollection
39     pressure = random.choice(pressureCollection)
40     print("Pressure: ", pressure, "hpa")
41     return pressure
42
43 def calculate_speed_kmh(time_sec):
44     global count
45     circumference_cm = (2 * math.pi) * radius_cm
46     rotations = count / 2.0
47
48     dist_km = (circumference_cm * rotations) / CM_IN_A_KM
49
50     km_per_sec = dist_km / time_sec
51     km_per_hour = km_per_sec * SECS_IN_AN_HOUR
52
53     speed = km_per_hour * ADJUSTMENT
54
55     return speed
56
57 def spin():
58     global count
59     count = count + 1
60     print(count)
61
62 # Gpio4 auf dem GrovePi+
63 wind_speed_sensor = DigitalInputDevice(4)
64 wind_speed_sensor.when_activated = spin
65
66 def readSensors():
67     global count
68     print("Method Started")

```

```

69     while True:
70         count = 0
71         sleep(interval)
72         [temperature, humidity] = grovepi.dht(temp_hum_sensor, blue)
73         pressure = simulatePressure()
74         wind_speed = calculate_speed_kmh(interval)
75         airtight = calculateAirtight(temperature, pressure)
76
77 # Konvertiere sensorData zu String fuer Sending request
78         sensorData = {
79             "temperature": str(temperature),
80             "wind_speed": str(wind_speed),
81             "pressure": str(pressure),
82             "humidity": str(humidity),
83             "airtight": str(airtight)
84         }
85         print("SensorData", sensorData)
86         print("SensorDataTypebefore sending", type(sensorData))
87
88         send.send_data(sensorData)
89         return sensorData
90
91 # Luftdichte berechnen
92 def calculateAirtight(temp_cel, pressure):
93
94     temp_cel = temp_cel
95     temp_kel = (temp_cel + 273.15)
96     p = pressure
97     r = 287.05
98
99     print("pressure", p)
100    print("r", r)
101    print("temp_kel", temp_kel)
102    print("temp_cel", temp_cel)
103
104    airtight = (p * 100) / (r * temp_kel)
105    airtight = round(airtight, 2)
106

```

```

107     print("Die Luftdichte bei ", temp_cel, "Grad Celcius und ", p, "hPa"
108     betraegt ", airtight, "kg/m**3")
109
110     return airtight
110 while True:
111     readSensors()

```

## B.2. Python Skript: send\_test\_data.py

```

1 # Daten senden via REST Edge Gateway
2 # CURL-Command
3 # curl -v -H "Content-Type: application/json" -d "{ \
4     capabilityAlternateId\": \"1\", \"sensorTypeAlternateId\": \"0\",
5     \"measures\": [{\"temperature\": \"25\"}] }"
6 # http://<IP_GW_EDGE>:<PORT_GW_EDGE>/measures/<DEVICE_ALTERNATE_ID>
7
8 from grovepi import *
9 import sys
10 import requests
11 import json
12 import time
13 import math
14 from datetime import datetime
15
16 # Variablen fur Verbindung mit IOT-Services
17
18 deviceAlternateId = '60b373db-d6dc-4696-b916-aefec6d95cd0'
19 sensorAlternateId = 'root'
20 capabilityAlternateId = '1234' # wind_1
21 tenant = 'https://5075f8b9-866e-4a4b-82f8-74687b72f1ab.eu10.cp.iot.sap/
22 iot/gateway/rest/measures/' # the IoT Service Host Name
23
24 postAddress = ('http://192.168.178.52:8699/measures/' +
25     deviceAlternateId)
26 postAddressCommand = ('http://192.168.178.52:8699/commands/' +
27     deviceAlternateId)
28 print('Posting to:', postAddress)

```

```

24
25 led = 3 # Digital Port 3
26 pinMode(led, "OUTPUT")
27 time.sleep(1)
28
29 # Methode zum Erhalten von Commands
30
31 def get_command():
32     print("Sending Command Started")
33
34     try:
35
36         headers = {'content-type': 'application/json'}
37         r = requests.get(postAddressCommand, deviceAlternateId)
38
39         responseCode = r.status_code
40
41         print("==> HTTP Response: %d" % responseCode)
42
43         result = r.text
44         print(result)
45
46         print(type(result))
47         result = result.encode('utf8')
48         print(type(result))
49         print(result)
50
51     except IOError:
52         print("Error")
53
54     ledtrue = "true"
55
56     if ledtrue in result:
57         digitalWrite(led, 1)      # Send HIGH to switch on LED
58         print("LED True!")
59         time.sleep(4)
60         digitalWrite(led, 0)      # Send HIGH to switch on LED
61         print("LED Off")
62         time.sleep(1)

```

```

63
64 # Methode zum Senden von Sensordaten an die Cloud
65 def send_data(sensorData):
66     print("Sending Method Started")
67     print("SensorDataType", type(sensorData))
68     try:
69         print("")
70         print("====")
71         print("Reading sensor data ...")
72
73         bodyJson = {
74             "capabilityAlternateId": capabilityAlternateId,
75             "sensorAlternateId": sensorAlternateId,
76             # "measures": [{"temperature": sensorData[0]}, {"wind_speed":
77             ": sensorData[1]}, {"pressure": sensorData[2]}, {"humidity": sensorData[3]}]
78             "measures": [{"temperature": sensorData["temperature"]}, {"wind_speed": sensorData["wind_speed"]}, {"pressure": sensorData["pressure"]}, {"humidity": sensorData["humidity"]}, {"airtight": sensorData["airtight"]}]
79         }
80
81         data = json.dumps(bodyJson)
82         headers = {'content-type': 'application/json'}
83         r = requests.post(postAddress, data=data, headers=headers)
84         responseCode = r.status_code
85         print(str(bodyJson))
86         print("==> HTTP Response: %d" % responseCode)
87
88     except IOError:
89         print("Error")
90
91     get_command()

```

### B.3. Python Skript: AWS\_Flask.py

```

1 import boto3

```

```

2 from flask import Flask
3 from flask import request
4 import os
5 import time
6 from sap.cf_logging import flask_logging
7 import logging
8
9 app = Flask(__name__)
10 flask_logging.init(app, logging.INFO)
11 logger = logging.getLogger('cf.dev.beta.bla.aws_sms')
12
13 cf_port = os.getenv("PORT")
14
15 time_last_sms = 0
16 time_curr = 0
17 diff_min = 0
18
19 @app.route('/')
20 def input():
21     logger.info('MOIN')
22     return("Ok..")
23
24 @app.route('/postjson', methods=['POST'])
25 def postJsonHandler():
26     global time_curr
27     global time_last_sms
28     global diff_min
29
30     time_curr = int(time.time()) # aktuelle zeit in unix epoch
31     timestamp
32     diff_min = round(((time_curr - time_last_sms) / 100), 0) # letzter
33     sendezeitpunkt minus aktuelle zeit
34
35     logger.info("last sms sent: " + str(time_last_sms))
36     logger.info("time_curr: " + str(time_curr))
37     logger.info("diff_min: " + str(diff_min))
38
39     if diff_min < 0 or diff_min > 2:
40         #print(request.is_json)

```

```

39     content = request.get_json()
40
41     #print(content)
42     phone = content['phone']
43     message = content['message']
44     #print("{}".format(phone))  # you can comment it out
45     #print("{}".format(message))  # you can comment it out
46
47     # Create an SNS client
48     client = boto3.client("sns",
49                             aws_access_key_id="AKIAI44Y3*****",
50                             aws_secret_access_key="EXKQJR2D9*****",
51                             region_name="eu-west-1"
52                         )
53
54     # Send your sms message.
55     logger.info('sending sms ...')
56     client.publish(
57         PhoneNumber="{}".format(phone),
58         Message="{}".format(message)
59     )
60     time_last_sms = time_curr # zeitpunkt zuletzt gesendet auf
61     aktuellen sendezzeitpunkt setzen
62     return 'SMS alert sent :)'
63     return 'Last SMS sent before 2min intervall.'
64 if __name__ == '__main__':
65     if cf_port is None:
66         app.run(host='0.0.0.0', port=5000, debug=True)
67     else:
68         app.run(host='0.0.0.0', port=int(cf_port), debug=True)

```

## B.4. API-Anfragen für Aktionen

```

1   # Aktion Event_Orkan
2   POST to Destination Standard_EventType: https://events-sap.cfapps.
eu10.hana.ondemand.com/ES/EventType/com.sap.appiot.eventtypes:

```

```

    StandardEventType/v1/Events
3     {
4     "BusinessTimestamp": "${event.time}",
5     "Type": "Alert",
6     "EventInfo": "Alert on Wind_Speed",
7     "EventStatus": "Open",
8     "EventSeverity": 1,
9     "EventCode": null,
10    "EventSource": null,
11    "ThingId": "${thing.id}",
12    "ThingProperty": "wind_speed",
13    "ExternalId": "Orkan"
14  }
15  # Aktion Event_Sturm
16  POST to Destination Standard_EventType: https://events-sap.cfapps.eu10.hana.ondemand.com/ES/EventType/com.sap.appiot.eventtypes:
    StandardEventType/v1/Events
17  {
18    "BusinessTimestamp": "${event.time}",
19    "Type": "Alert",
20    "EventInfo": "Alert on Wind_Speed",
21    "EventStatus": "Open",
22    "EventSeverity": 2,
23    "EventCode": null,
24    "EventSource": null,
25    "ThingId": "${thing.id}",
26    "ThingProperty": "wind_speed",
27    "ExternalId": "Sturm"
28  }
29  # Aktion Event_Temperature
30  POST to Destination Standard_EventType: https://events-sap.cfapps.eu10.hana.ondemand.com/ES/EventType/com.sap.appiot.eventtypes:
    StandardEventType/v1/Events
31  {
32    "BusinessTimestamp": "${event.time}",
33    "Type": "Alert",
34    "EventInfo": "Alert on engine temperature",
35    "EventStatus": "Open",
36    "EventSeverity": 1,

```

```
37  "EventCode": null,  
38  "EventSource": null,  
39  "ThingId": "${thing.id}",  
40  "ThingProperty": "temperature",  
41  "ExternalId": "Kritische Temperatur"  
42 }
```

## Abschließende Erklärung

Ich versichere hiermit, dass ich meine Masterarbeit selbständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

<ORT>, den 28. Januar 2020

<AUTOR>

## C. Notizen, die ich später noch gebrauchen könnte

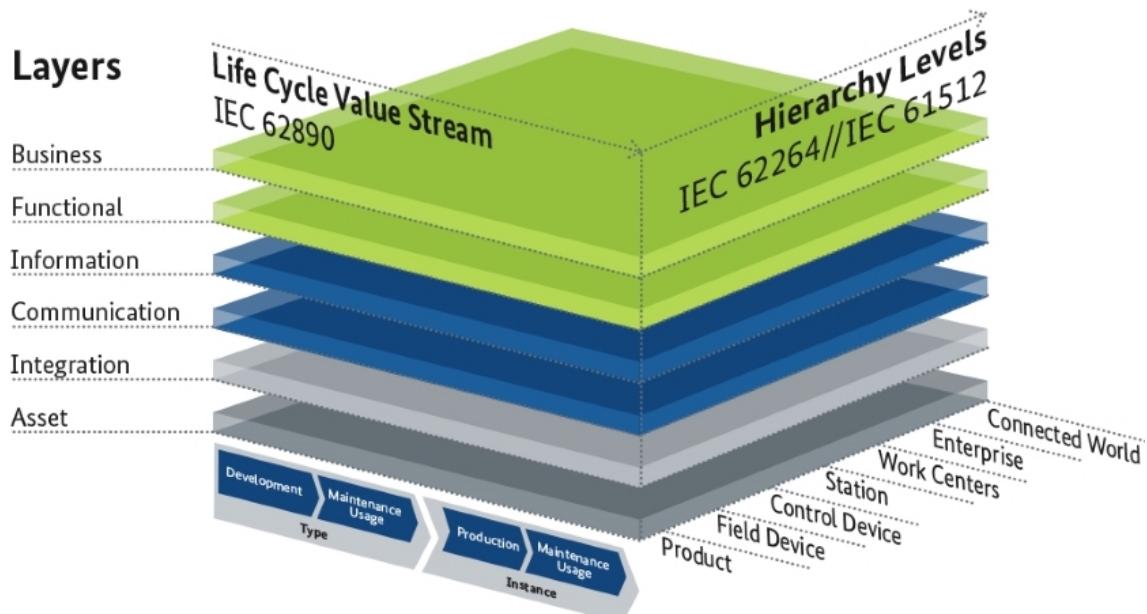


Abbildung 27: Das Referenzarchitekturmodell Industrie 4.0 (Bitkom e.V., 2015)

## Die Industrie-4.0-Komponente

- Bitkom bzw. Plattform Industrie 4.0 stellt Referenzarchitektur
- RAMI bietet die Möglichkeit, Industrie 4.0 UseCases zu verorten, um die für den jeweiligen use case notwendigen Normen und Standards zu identifizieren
- horizontale Integration über Wertschöpfungsnetzwerke: Lieferanten, Unternehmen, Produzenten
- Durchgängigkeit des Engineerings: Systems Engineering, Modellierung, Simulation
- vertikale Integration und vernetzte Produktionssysteme mit Echtzeitanforderung
- neue soziale Infrastrukturen der Arbeit: Humane-Machine-Systeme und Usability
- kontinuierliche Entwicklung von Querschnittssystemen: Netzkomunikation, Breitband, Cloud, Data Analytics, Cyber security

(Industrie 4.0 Weltweit)

Das RAMI 4.0 basiert auf den Grundideen des Smart Grids, welches das Stromnetz von der Erzeugung bis zur Verteilung zum Endverbraucher behandelt. Das dreidimensionale Modell kapselt die wichtigsten Funktionalitäten in mehreren Schichten. Dies schafft Flexibilität für die Konzeptionisierung und Realisierung von Industrie 4.0-Lösungen.

- Aspekte Industrie 4.0 (Bitkom S. 40) -> Für diese Ziele muss eine Referenz geschaffen werden
- 1 - vertikale Integration innerhalb der Fabrik/der Produktion: Vernetzung von Produktionsmitteln wie Automatisierungsgeräte oder Dienste untereinander
- 2 - Einbeziehung der Produkte (in meinem Fall produzierte Energie)
- durchgängiges Engineering bedeutet: technische, administrative, kommerzielle Daten rund um das Produktionsmittel über Wertschöpfungskette hinweg konsistent halten und jederzeit über das Netzwerk zugreifbar machen

- 3 - horizontale Integration über Wertschöpfungsnetzwerke über den Fabrikstandort hinaus und dynamische Bildung von Wertschöpfungsnetzwerken
- es gibt RAMI 4.0
- und Referenzmodell für die Industrie 4.0-Komponente (s. 45) mit Verwaltungsschale

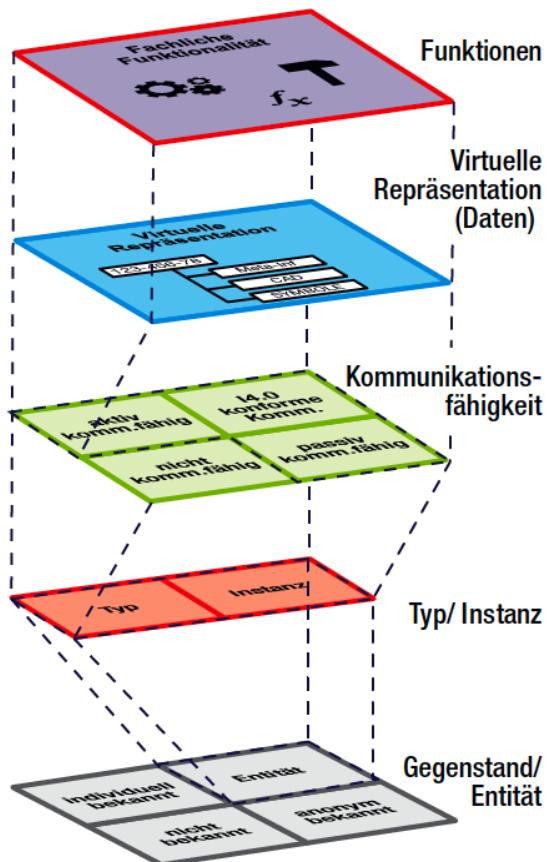


Abbildung 28: Ebenen der Industrie-4.0-Komponente (Bitkom e.V, 2015, S. 52)