



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Die Steuerung von digitalen Zwillingen mit
Workflow-Management-Systemen am Beispiel einer virtuellen
Fertigungslinie“

verfasst von / submitted by

Lorenz Pircher BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA 066 926

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Masterstudium Wirtschaftsinformatik

Betreut von / Supervisor:

Univ.-Prof. Mag. Dipl.-Ing. Dr.techn.
Maria Leitner, Bakk

Danksagung

Ein besonderer Dank gebührt meiner Betreuerin Univ.-Prof. Mag. Dipl.-Ing. Dr.techn. Maria Leitner, Bakk, welche stets ansprechbar war und mir mit zahlreichen Meetings und Ratschlägen das Schreiben meiner Masterarbeit deutlich erleichtert hat.

Ebenfalls möchte ich ao. Univ.-Prof. tit. Univ.-Prof. Dipl.-Ing. Dr. Erich Schikuta danken, welcher spontan als Gutachter einspringen konnte.

An dieser Stelle möchte ich mich noch bei meiner Familie, Freunden sowie meiner Partnerin bedanken, die mich auf meinem Weg stets unterstützt haben.

Lorenz Pircher

Abstract

Digital twins are considered to be essential for the realization of Industry 4.0 and are regarded by many experts as a key technology for this. They enable the retrieval of real-time data, the digital planning of services or productions, the execution of virtual experiments and simulations as well as process optimizations. However, due to a lack of standards, frameworks, regulations and other challenges, the concept of digital twins is still in the start-up phase and remains a largely theoretical concept. Parallels can be drawn to workflow systems, which are processes that are either fully or partially automated by a software system, given that the implementation of business process management can also yield process optimizations, including potential cost reductions, shorter execution times and reduced error rates.

This thesis aims to investigate how digital twins and Workflow Management Systems (WfMS) can be combined and interconnected. The methodology includes an extensive literature review on the background of both technologies, highlighting in particular application areas where those technologies complement each other. Furthermore, a concept as well as an implementation approach is presented, which allows the realization of two concrete use cases of a manufacturing plant. This implementation allows to control a process-based digital twin with the help of a WfMS. Among other things, this digital twin can perform 3D-simulations and prepare real-time data in the form of a dashboard. In addition, a web application of the digital twin provides users with an intuitive interface to define, manage and run simulations. These simulations are controlled by the WfMS, which in turn controls the digital twin. The presented concept enables an improved collaboration, even for non-experts, as the underlying processes of the digital twin are available in a generally understandable notation and can be easily adapted.

Keywords— Digital Twin, Industry 4.0, Process Automation, Internet of Things, Workflow System, Smart Manufacturing & Smart Factory

Kurzfassung

Digitale Zwillinge gelten als essenziell für die Realisierung der Industrie 4.0 und werden von vielen Experten als Schlüsseltechnologie dieser angesehen. Sie ermöglichen das Abrufen von Echtzeitdaten, die digitale Planung von Diensten oder Produktionen, die Durchführung virtueller Experimente und Simulationen sowie Prozessoptimierungen. Aufgrund fehlender Standards, Frameworks, Regulierungen und weiterer Herausforderungen befindet sich das Konzept der digitalen Zwillinge jedoch noch in der Startphase und bleibt ein größtenteils theoretisches Konzept. Hierbei können Parallelen zu Workflow Systemen, mittels Softwaresystem vollständig oder zum Teil automatisierte Prozesse, gezogen werden, da der Einsatz von Geschäftsprozessmanagement ebenso Prozessoptimierungen, wie mögliche Kostensenkungen, Verkürzungen der Durchführungszeiten sowie Verringerungen der Fehlerquote mit sich bringen kann.

Diese Abschlussarbeit zielt darauf ab, zu untersuchen, wie digitale Zwillinge und Workflow-Management-Systeme (WfMS) gemeinsam genutzt und miteinander vernetzt werden können. Die Methodik umfasst eine umfangreiche Literaturrecherche zu den Hintergrundinformationen beider Technologien, wobei insbesondere Anwendungsbereiche beleuchtet werden, in denen sich jene Technologien ergänzen. Darüber hinaus wird ein Konzept sowie ein Implementierungsansatz vorgestellt, welche die Durchführung zweier konkreter Anwendungsfälle einer Fertigungsanlage ermöglichen. Jene Implementierung erlaubt es, einen prozessbasierten digitalen Zwilling mithilfe eines WfMS zu steuern. Dieser digitale Zwilling kann unter anderem 3D-Simulationen durchführen sowie Echtzeitdaten in Form eines Dashboards aufbereiten. Zudem wird eine Web-Anwendung entwickelt, welche Benutzern eine intuitive Schnittstelle bietet, um Simulationen zu definieren, zu verwalten und auszuführen. Diese Simulationen werden vom WfMS gesteuert, welches wiederum den digitalen Zwilling kontrolliert. Das vorgestellte Konzept ermöglicht eine verbesserte Zusammenarbeit, auch für Nicht-Experten, da die zugrundeliegenden Prozesse des digitalen Zwillings in einer allgemein verständlichen Notation vorliegen und einfach angepasst werden können.

Schlüsselbegriffe— Digitaler Zwilling, Industrie 4.0, Prozessautomatisierung, Internet der Dinge, Workflow System, Smart Manufacturing & Smart Factory

Inhaltsverzeichnis

Danksagung	i
Abstract	iii
Kurzfassung	v
Tabellenverzeichnis	xi
Abbildungsverzeichnis	xiii
Codeausschnitte	xv
1. Einleitung	1
2. Verwandte Arbeiten	3
3. Ziele und Forschungsfragen	5
4. Hintergrund: Digitaler Zwilling	7
4.1. Definition	8
4.2. Ursprung	9
4.3. Treibertechnologien	10
4.3.1. Konnektivität	11
4.3.2. Digitalisierung	12
4.3.3. Intelligenz	12
4.4. Rolle digitaler Zwillinge in der Industrie 4.0	13
4.4.1. Smart Manufacturing	14
4.4.2. Smart Factory	15
4.4.3. Unterschied zwischen digitalem Zwilling und Simulation	16
4.5. Anwendung	17
4.5.1. Anwendung in Fabriken	17
4.5.2. Anwendung in Smart Cities	19
4.6. Arten digitaler Zwillinge	19
4.7. Elemente eines digitalen Zwillings	21
4.8. Architektur und Anforderungen	22
4.9. Herausforderungen	23
4.10. Digitaler Zwilling Software	23
4.10.1. NVIDIA OVX	24

4.10.2. Siemens Lösungen	24
4.10.3. Eclipse Ditto	25
4.10.4. AWS IoT	25
4.10.5. Azure Digital Twins	25
4.10.6. FlexSim	25
4.10.7. Factory I/O	26
4.11. Digitale Zwillinge in der Zukunft	26
4.12. Zusammenfassung	26
5. Hintergrund: Workflow-Management-System	27
5.1. Geschäftsprozessmanagement	27
5.2. Business Process Model and Notation (BPMN)	27
5.3. Workflow und WfMS	27
5.4. WfMS Architektur	29
5.5. WfMS Software	31
5.5.1. AristaFlow BPM Suite	31
5.5.2. ADONIS GPM-Suite	32
5.5.3. Bonita	32
5.5.4. Cloud Process Execution Engine (CPEE)	32
5.6. Workflow Dimensionen	33
5.7. Workflow Verifikation	33
5.7.1. Petri-Netze	33
5.7.2. Workflow Netze (WF-Nets)	34
5.8. Ausblick: Process Mining	36
5.9. Zusammenfassung	36
6. Gemeinsamer Einsatz digitaler Zwillinge und WfMS	37
WfMS und digitale Zwillinge einer Organisation	37
Modellgesteuerte PAISs	37
Modellgesteuerter digitaler Zwilling	38
Prozessbewusstes digitales Zwillingscockpit (PADTC)	38
Entscheidungsfindungsmodell eines digitalen Zwillings	39
Rückgrat eines digitalen Zwillings	39
7. Methodik	41
8. Konzept	43
8.1. Konzeptuelle Steuerung eines digitalen Zwillings anhand eines WfMS . . .	44
8.2. Entwurf der Digital Twin Simulation Application	44
8.3. Weitere Aspekte	46
9. Technische Implementierung	47
9.1. Umsetzung des Prototyps	49
9.1.1. Architektur der Implementierung	50

9.1.2.	Simulations-API	53
9.1.3.	Digital Twin Simulation Application (TWIN)	53
9.1.4.	Steuerung des digitalen Zwillings durch das WfMS	53
9.1.5.	Automatisierung der Fertigungslinie	54
9.1.6.	End-to-End IoT Integration	55
9.2.	Eingesetzte Programme und Technologien	57
9.2.1.	OpenPLC	57
9.2.2.	Factory I/O	58
9.2.3.	Cloud Process Execution Engine als WfMS	59
9.2.4.	Echtzeitdatenbank und Dashboard	59
9.2.5.	Swagger UI	60
9.2.6.	Kafka und MQTT	61
9.3.	Aufbau von TWIN	61
9.3.1.	Unterseite: Healthcheck	62
9.3.2.	Unterseite: Simulation erstellen	62
9.3.3.	Unterseite: Geräte Übersicht	63
9.3.4.	Unterseite: Simulation durchführen	64
9.3.5.	Unterseite: Dashboard	65
9.4.	Installations- und Konfigurationsanleitung	66
9.4.1.	Systemvoraussetzungen und Tools	66
9.4.2.	Zugriff und Ausführung des Codes	66
9.4.3.	Simulations-Logs in der Konsole	71
9.5.	Einblicke in den Code	71
9.6.	API Beschreibung	76
9.7.	Limitierungen	77
10.	Anwendungsfälle	79
10.1.	Anwendungsfall 1: Produktion von Möbel	79
10.2.	Anwendungsfall 2: Verpackung und Versand	83
10.3.	Überprüfung der WF-Net Eigenschaften	83
11.	Evaluierung	87
11.1.	Anwendungsfall 1: Produktion von Möbel	90
11.2.	Anwendungsfall 2: Verpackung und Versand	92
11.3.	Ausführung einer prozessbasierten Simulation	94
12.	Conclusio	97
12.1.	Ausblick	99
	Abkürzungsverzeichnis	103
	Literaturverzeichnis	105
A.	Anhang	117

Tabellenverzeichnis

4.1. Verschiedene Definitionen des digitalen Zwillings	9
4.2. Gegenüberstellung digitaler Zwilling-Software	24
5.1. Gegenüberstellung WfMS-Software	31
9.1. Microservice UI URL-Adressen	70
A.1. Übersicht der verwendeten Tags in Factory I/O	117
A.2. Petri-Netz Bezeichnungen Anwendungsfall 1: Produktion von Möbel . . .	125
A.3. Petri-Netz Bezeichnungen Anwendungsfall 2: Verpackung und Versand . .	126

Abbildungsverzeichnis

2.1. Anzahl gefundener Publikationen via DBLP	4
4.1. Visualisierung eines digitalen Zwillings	7
5.1. Transformation eines Prozesses in ein Prozessmodell eines WfMS	28
5.2. Architektur von Workflows (Nachbildung)	30
6.1. Digitales Zwillingscockpit (Nachbildung)	39
8.1. Konzept zur Steuerung eines digitalen Zwillings anhand eines WfMS . . .	43
8.2. Anwendungsfalldiagramm der TWAIN	45
9.1. Abstrakte Darstellung der Softwarearchitektur	47
9.2. Detaillierte Darstellung der Softwarearchitektur	52
9.3. Sequenzdiagramm: Simulation erstellen und durchführen	54
9.4. End-to-End IoT Integration in der Implementierung	56
9.5. FUP Ausschnitt in OpenPLC (Screenshot)	58
9.6. TWAIN Startseite (Screenshot)	61
9.7. TWAIN Healthcheck (Screenshot)	62
9.8. Formular der TWAIN zur Definition von Simulationen (Screenshot) . . .	63
9.9. TWAIN Geräte Übersicht (Screenshot)	64
9.10. Durchführung von WfMS-Instanzen in der TWAIN (Screenshot)	64
9.11. Dashboard des digitalen Zwillings in der TWAIN (Screenshot)	65
9.12. Factory I/O Modbus Server Adresse (Screenshot)	68
9.13. Docker-Desktop nach dem Start der Implementierung (Screenshot) . . .	70
9.14. Simulations-Logs in der Konsole (Screenshot)	71
9.15. Simulations-API Beschreibung in Swagger (Screenshot)	77
10.1. Anwendungsfall 1: Produktion von Möbel (BPMN-Diagramm)	81
10.2. Anwendungsfall 2: Verpackung und Versand (BPMN-Diagramm)	82
10.3. Anwendungsfall 1: Produktion von Möbel als WF-Net	85
10.4. Anwendungsfall 2: Verpackung und Versand als WF-Net	86
11.1. Überblick: Gesamte Fertigungslinie in Factory I/O (Screenshot)	88
11.2. Startpunkt der Fertigungslinie in Factory I/O (Screenshot)	89
11.3. Schrank-Produktion der Fertigungslinie in Factory I/O (Screenshot) . . .	90
11.4. Schreibtisch-Produktion der Fertigungslinie in Factory I/O (Screenshot) .	91

Abbildungsverzeichnis

11.5. Treffpunkt beider Produktionen der Fertigungsline in Factory I/O (Screenshot)	92
11.6. Verpackung und Versand in Factory I/O (Screenshot)	93
11.7. Anwendungsfälle in CPEE (Screenshot Teil 1/2)	95
11.8. Anwendungsfälle in CPEE (Screenshot Teil 2/2)	96

Codeausschnitte

9.1. Docker-Compose starten (Terminal)	67
9.2. .env File anpassen	68
9.3. mbconfig.cfg File erstellen bzw. anpassen	68
9.4. Erstellung des auto.cfg Files	69
9.5. SKD von Factory I/O starten (Terminal)	70
9.6. Implementierung mit einem Befehl starten (Terminal)	70
9.7. RFID-Leser Klasse (C#)	72
9.8. MQTT Topic via SDK senden (C#)	73
9.9. Task einleiten (NodeJS)	74
9.10. Task beenden (NodeJS)	75
9.11. Kafka Topic in InfluxDB speichern (NodeJS)	76

1. Einleitung

Digitale Zwillinge sind eine der vielversprechendsten Schlüsseltechnologien in der heutigen Zeit der Digitalisierung. Nach der Definition von Kuhn [1] ist ein digitaler Zwilling eine digitale Repräsentanz eines Dinges aus der realen Welt. Im Vergleich zu anderen existierenden Definitionen sei es in diesem Zusammenhang irrelevant, ob das Gegenstück in der realen Welt bereits existiere oder erst in der Zukunft verwirklicht wird. Mit Hilfe eines digitalen Zwillings sollen alle relevanten Informationen sowie Dienste von Dingen via einer einheitlichen Schnittstelle zur Verfügung gestellt werden. Auf diese Weise soll ein übergreifender Informationsaustausch realisiert werden, welcher für jene neuen Möglichkeiten, die sich durch digitale Zwillinge ergeben, maßgeblich ist. So können anhand eines digitalen Zwillings unter anderem Echtzeitdaten abgerufen, Dienste oder Produktionen digital geplant, virtuelle Experimente und Simulationen durchgeführt sowie Prozessoptimierungen ermöglicht werden. Laut Kuhn [1] könne demnach eine geplante Produktionsanlage bereits vor jener Inbetriebnahme über einen digitalen Zwilling verfügen. Weiters bestehe die Möglichkeit, digitale Zwillinge für Tests und Zertifizierungen einzusetzen, welche andernfalls mit immensen Kosten verbunden wären. Zusätzlich zu Echtzeitdaten, können digitale Zwillinge auf historische Daten zugreifen, wodurch beispielsweise Gründe für Defekte aufgedeckt und zukünftig vermieden werden können [1]. Insbesondere, aufgrund neuer Vorteile hinsichtlich Produkt- sowie Prozessoptimierung sind digitale Zwillinge eine essenzielle Schlüsseltechnologie für die Industrie 4.0 sowie für die digitale Fertigung [2]. So soll durch die Anwendung digitaler Zwillinge eine cyber-physische Integration in der Fertigung erreicht werden. Hierbei liegt der Fokus auf der Realisierung einer intelligenten Fertigung (engl.: Smart Manufacturing) [3]. Wichtige Möglichmacher für digitale Zwillinge sind aktuelle Technologien wie das Internet der Dinge, engl.: Internet of Things (IoT), Big Data, Cloud Computing, Künstliche Intelligenz, engl.: Artificial Intelligence, AI (KI) sowie weitere Informationstechnologien der neusten Generation. Der heutige Trend besteht darin, immer mehr Geräte mit dem Internet zu verbinden. Dementsprechend bietet das Abrufen aktueller Daten solcher Geräte, das Generieren und Speichern jener Datenmengen sowie die Organisation, Analyse und Auswertung wertvolle Chancen für viele Branchen, insbesondere der Produktion [3]. Bei Letzterer erfordert die Errichtung einer Produktionslinie generell jede Menge Zeit und Ressourcen. Aufgrunddessen versucht man mit Hilfe von Workflow Systemen, sprich mittels Softwaresystemen voll- oder teil-automatisierte Prozesse, jenen Prozess zu beschleunigen, indem das System entworfen und getestet wird, noch bevor eine Anlage gekauft oder in Betrieb genommen wird [4]. Die Automatisierung eines Prozesses bzw. eines Teiles davon dient unter anderem der Dokumentation, Optimierung und Standardisierung [5].

2. Verwandte Arbeiten

Um den aktuellen Forschungsstand widerzuspiegeln, wurden in einer vorhergegangenen Literaturrecherche die beiden Themengebiete digitale Zwillinge sowie Workflow Systeme und deren Anwendung aufgearbeitet. Die Recherche ergab, dass digitale Zwillinge in den unterschiedlichsten Einsatzgebieten angewendet werden. Hinsichtlich dieser Arbeit lag der Fokus insbesondere auf der Industrie, genauer gesagt der Fertigung. Ziel der Recherche war es, herauszufinden, ob jene beiden Technologien in irgendeiner Form, sei es in der Theorie, als auch Praxis, bereits gemeinsam zum Einsatz kommen. Aus der Recherche ging hervor, dass in den vergangenen Jahren zahlreiche wissenschaftliche Publikationen hinsichtlich digitaler Zwillinge und Workflow Systemen veröffentlicht wurden. Eine Google Suche ergab, dass der Gedanke an einem gemeinsamen Einsatz digitaler Zwillinge und Workflow Systemen nicht neu ist. Dies zeigen existierende Geschäftsprozess-Tools wie Signavio und Adonis, welche die Erstellung digitaler Zwillinge einer Organisation (engl.: Digital Twin of an Organization, DTO) ermöglichen [6, 7]. Aus der Recherche ging hervor, dass die Fertigung zu den häufigsten Anwendungsgebieten digitaler Zwillinge zählt. Laut Siemens [8] diene ein digitaler Produktionszwilling der Produktionsplanung, deren Ziel darin bestehe, sicherzustellen, dass die Produktion die erforderliche Leistung erbringt. Der Zwilling umfasse dabei alles von der Programmierung bis hin zur Automatisierungshardware und ermögliche anhand einer virtuellen Inbetriebnahme die Optimierung von Produktionen noch bevor Fertigungsanlagen in der realen Welt umgesetzt werden. Mit Hilfe des Zwillings können gesamte Fertigungsprozesse in einer vollständig virtuellen Umgebungen geplant werden, wodurch die Faktoren Zeit, Aufwand und Risiko für die bevorstehende Inbetriebnahme reduziert werden können. Anhand des Zwillings können Fertigungslinien-Layouts entworfen, Materialflüsse visualisiert, mögliche Engpässe ermittelt sowie speicherprogrammierbare Steuerungen (SPS), weithin bekannt als Programmable Logic Controller (PLC), simuliert werden [8]. Laut Microsoft [9] umfasse ein Prozesszwilling, eine weitere Art eines digitalen Zwillings, eine gesamte Produktionsumgebung. Jene Zwillinge bieten virtuelle Nachbildungen des Geschehens in Fabrikhallen und verhelfen mittels moderner Technologien gesamte Fertigungsprozesse zu optimieren. Weiters unterstütze die Art eines digitalen Zwillings die Zusammenarbeit zwischen Menschen und Maschinen [9]. Die meisten gefundenen wissenschaftlichen Artikel befassen sich zwar nicht direkt mit digitalen Zwillingen und Workflow Systemen, dafür aber zumindest mit verwandten Themengebieten beider Technologien. Rabah et al. [10] präsentieren das Konzept des digitalen Zwillings für Services (engl.: Digital Twin for Services, DT4S) und beschreiben ein Workflow-Management-System (WfMS) als Rückgrat sowie zentrale Komponente digitaler Zwillinge. Anhand jener können digitale Funktionen und Workflows automatisiert und reibungslos ausgeführt werden [10]. Rinderle-Ma und Mangler [11] befassen sich mit mehreren verwandten Themengebieten von digitalen Zwillingen und Workflow Systemen.

2. Verwandte Arbeiten

Hierzu zählen unter anderem Prozessautomatisierung, Sensoren, Process Mining sowie Fertigung. Im Rahmen dieser Arbeit beantworten Rinderle-Ma und Mangler [11] die Frage, wie Fertigungsprozesse automatisiert werden können. Weiters heben die Autoren hervor, dass Maschinen während der gesamten Laufzeit beobachtbar sein müssen, da dies maßgeblich für zukünftig stattfindende Datenanalysen zur Prozessoptimierung sei. Außerdem könne durch die Anbringung weiterer Sensoren, welche nicht wesentlich für die Produktion sind, Kontext geschaffen werden. Im Rahmen dieser Arbeit wurden 16 Fertigungsprozesse in der Cloud Process Execution Engine (CPEE) realisiert, welche Vorteile wie Rückverfolgbarkeit, Flexibilität und Nachhaltigkeit für die Unternehmen mit sich brachten [11]. Maamar et al. [12] präsentieren einen Ansatz zur Integrierung von IoT in Geschäftsprozesse. Hierzu erweitert man traditionelle Workflow-Sprachen mit Konstrukten wie beispielsweise die Rolle eines Dinges. Weiters führen die Autoren einen sogenannten Prozess der Dinge (engl.: Process of Things, PoT) ein. Ein PoT habe eine Geschichte, solche können in Form eines Skripts von Dingen *gespielt* werden. Zusätzlich verfügt ein PoT über Charaktere, die Dinge verkörpern und Szenen (auch Orte genannt), wo Dinge agieren. Zur Veranschaulichung präsentieren Maamar et al. [12] eine Implementierung eines Krankenhausszenarios, in welchem sie von Sensoren, verschiedenen IoT-Technologien sowie Kommunikationsprotokollen Gebrauch machen. In Bezug auf die Realisierung der bevorstehenden Masterarbeit sind diese besonders interessant, da jene genannten Technologien bei dieser ebenso zum Einsatz kommen sollen [12]. Hinsichtlich Software konnte anhand einer Google Recherche festgestellt werden, dass das Verhältnis zwischen kommerzieller Software und Open Source Programmen bis dato sehr ungleich ist. Neben *Eclipse Ditto*, einer digitalen Zwillings Software, welche in einer Open Source Architektur von Kamath, Morgan und Ali [13] Anwendung findet, ist auch *OpenPLC* eine der wenigen Open Source Alternativen, welche zur Automatisierung in der Industrie eingesetzt wird. Die Automatisierung solle die Qualität und Produktion steigern sowie Kosten senken [14]. Alves et al. [14] beschreiben die Erstellung eines kostengünstigen PLC mit Hilfe von OpenPLC. Abbildung 2.1 enthält gefundene Publikationen, die sich sowohl mit verwandten Themengebieten des digitalen Zwillings sowie WfMS befassen.

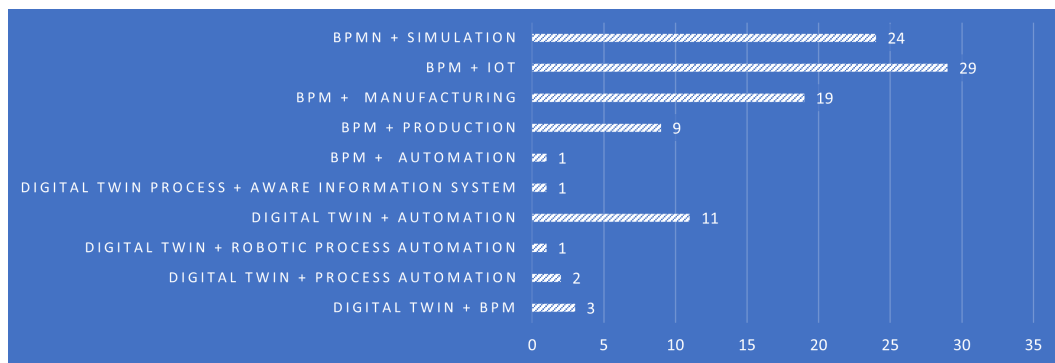


Abbildung 2.1.: Anzahl gefundener Publikationen via DBLP¹

3. Ziele und Forschungsfragen

Das Ziel dieser Abschlussarbeit ist die Steuerung von digitalen Zwillingen mit WfMS anhand einer virtuellen Fertigungslinie zu untersuchen sowie eine solche zu realisieren. Die erwartete Implementierung soll schließlich in der Lage sein, einen digitalen Zwilling einer Fertigungslinie mittels eines WfMS anzusteuern sowie folglich diesen zu kontrollieren. Auf diese Weise soll schließlich die Fertigung individueller Produkte in einer 3D-Simulationsumgebung durchgeführt werden können. Aus dem Fertigungsprozess resultierende Echtzeitdaten sollen in Form eines Dashboards kontinuierlich angezeigt werden. Zusätzlich ist die Ansammlung eines persistenten Datenbestandes für zukünftige Analysen erforderlich. Weiters soll erforscht werden, wie digitale Zwillinge und WfMS üblicherweise gemeinsam eingesetzt werden. Anhand einer Literaturrecherche, einer technischen Implementierung sowie der daraus resultierenden Resultate sollen schließlich die folgenden Forschungsfragen der Masterarbeit beantwortet werden können. Diese lauten wie folgt:

F1: Wie werden digitale Zwillinge in Kombination mit WfMS genutzt?

F2: Wie kann ein digitaler Zwilling anhand eines WfMS gesteuert werden?

F3: Wie wird ein möglicher Ansatz zur Implementierung eines Prozesszwillings gestaltet?

F4: Welche Anwendungsfälle in Produktion und Logistik gibt es?

Die erste Frage (F1) soll anhand einer Literaturrecherche beantwortet werden. Dabei soll ausgiebig nach gemeinsamen Verwendungen digitaler Zwillinge und WfMS gesucht werden. Die Antwort auf die nächsten Frage (F2) soll aus einem neuen Konzept bestehen, welches näher erläutert soll, wie ein WfMS mit einem digitalen Zwilling interagieren und diesen steuern kann. Zur Beantwortung der dritten Frage (F3) wird ein technischer Lösungsansatz präsentiert, welcher unter anderem die Herangehensweise näher bringen sowie die Reproduzierbarkeit für interessierte Leser/innen ermöglichen soll. Schließlich soll am Beispiel zweier konkreter Fertigungsprozesse aufgezeigt werden, für welche Anwendungsfälle in Produktion und Logistik sich ein Prozesszwilling eignen kann (F4).

4. Hintergrund: Digitaler Zwilling

Laut dem amerikanischen Marktforschungsunternehmen Gartner gehören digitale Zwillinge zu den vielversprechendsten technologischen Trends (2017 und 2018) [15, 16]. Digitale Zwillinge stehen im engen Kontakt mit Schlüsselbegriffen und -technologien wie IoT, Big Data, KI, Data Mining und vielen weiteren. Vor allem hinsichtlich der Industrie 4.0 spielen digitale Zwillinge eine entscheidende Rolle. So soll anhand einer cyber-physischen Integration in der Fertigung ein wichtiger Schritt für den Umstieg zu Smart Manufacturing gemacht werden können [3]. McKinsey [17] definiert einen digitalen Zwilling als eine virtuelle Darstellung eines physischen Objekts, einer Person oder eines Prozesses. Dieser Zwilling bestehe aus Daten unterschiedlicher Quellen, Visualisierungen sowie einer Schicht aus Verhaltenseinblicken, die mit Hilfe angesammelter Daten abgeleitet werden. Ein digitaler Zwilling bildet ein Ding nach und ermöglicht anspruchsvolle Analysen sowie fortschrittliche Simulationen.

Abbildung 4.1 zeigt dementsprechend auf der linken Seite ein reales Objekt, dessen Daten auf der Seite des digitalen Zwillings zur Visualisierung oder zur Ableitung von Verhaltenseinblicken genutzt werden können. Weiters können dem digitalen Zwilling Echtzeitinformationen entnommen werden.

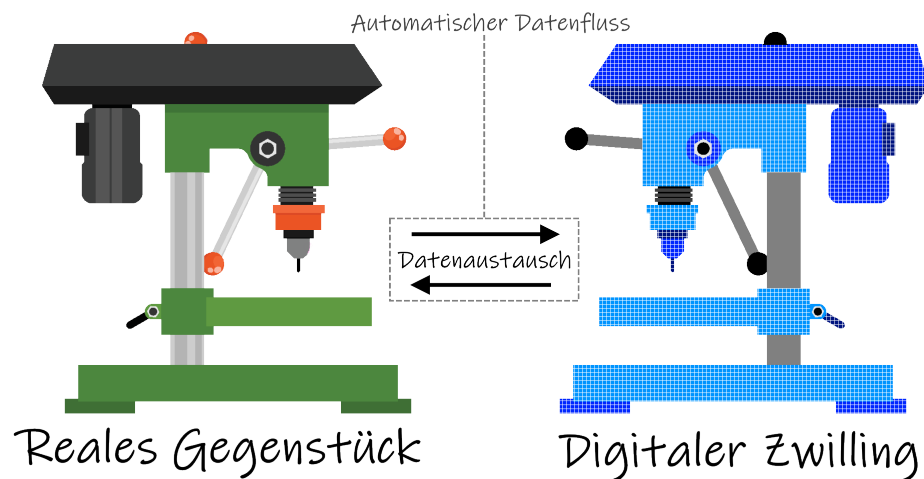


Abbildung 4.1.: Visualisierung eines digitalen Zwillings

4. Hintergrund: Digitaler Zwilling

Zur Erstellung digitaler Zwillinge einzelner Produkte werden moderne Technologien wie beispielsweise KI eingesetzt. Auf diese Weise versuchen Hersteller die Fähigkeiten solcher sowie die Leistung über den gesamten Lebenszyklus zu verbessern. Laut Microsoft [9] können Hersteller anhand einfacher digitaler Zwillinge nicht gleichzeitig am digitalen sowie physischen Modell eines Produktes arbeiten. Demnach seien keine Anpassungen in Echtzeit vornehmbar, was zu Verzögerungen und Stillständen bei erforderlichen Produktionsänderungen führe. Weiters ermöglichen konventionelle digitale Zwillinge keine Zusammenarbeit in der virtuellen Umgebung. In weiterer Folge präsentiert Microsoft den digitalen Zwilling eines Prozesses (engl.: Process Digital Twin), welcher der nächste Schritt der digitalen Transformation sei und einen positiven Effekt für die gesamte Fabrik und Lieferkette biete. Jener digitale Zwilling findet insbesondere in fortschrittlichen Fertigungsszenarien seine Anwendung, da dieser auf der Ebene von Maschinen, Fabriken und Lieferketten agiert [9].

4.1. Definition

Laut Köhler [18] liefere die Google-Suche für „Digitaler Zwilling“ sowie „Digital Twin“ insgesamt weitaus mehr als 300 Millionen Ergebnisse. Der Großteil dieser Ergebnisse sei deutschsprachig. Dabei hebt Köhler hervor, dass es so etwas wie eine einheitliche, verbindliche Definition für einen digitalen Zwilling nicht gebe. Die wesentlichen Unterschiede der einzelnen Definitionen seien verschieden gesetzte Schwerpunkte [18]. Die vorliegende Arbeit gebraucht die Definition von Kuhn [1]. Diese lautet wie folgt:

„Digitale Zwillinge sind digitale Repräsentanzen von Dingen aus der realen Welt. Sie beschreiben sowohl physische Objekte als auch nicht physische Dinge wie z.B. Dienste, indem sie alle relevanten Informationen und Dienste mithilfe einer einheitlichen Schnittstelle zur Verfügung stellen. Für den digitalen Zwilling ist es dabei unerheblich, ob das Gegenstück in der realen Welt schon existiert oder erst existieren wird [1, S. 440–444].“

Tabelle 4.1 umfasst eine chronologische Auflistung weiterer Definitionen digitaler Zwillinge, welche aufzeigen soll, inwiefern sich solche Definitionen im Laufe der Zeit entwickelt haben.

Jahr	Definition
2020	Das Konzept digitaler Zwillinge bezieht sich auf eine virtuelle Darstellung von Fertigungselementen wie Personal, Produkte, Anlagen und Prozesse. Dabei stellen digitale Zwillinge ein lebendes Modell dar, welches kontinuierlich aktualisiert und bei Änderungen des Gegenstücks verändert wird. So werden Status, Arbeitsbedingungen sowie Ressourcenzustände synchron dargestellt (Lu et al. [19]).
2019	Digitale Zwillinge sind die Schlüsseltechnologie zur Realisierung eines cyber-physischen Systems. Dabei wird ein virtueller Zwilling einer physischen Einheit erstellt, um die Eigenschaften einer physischen Einheit durch Simulation in Echtzeit zu optimieren und zu steuern (Ding et al. [20]).

2019	Digitale Zwillinge definieren ein integriertes System, anhand welches der Systemstatus sowie Prozesse überwacht, berechnet, reguliert und gesteuert werden können (Zheng et al. [21]).
2018	Ein digitaler Zwilling besteht aus den folgenden fünf Teilen, welche vorhanden sein müssen, damit der digitale Zwilling als vollständig angesehen wird: einen physischen Teil, einen virtuellen Teil, eine Verbindung jener Teile, Daten sowie Dienste (Tao et al. [22]).
2018	Digitale Zwillinge sind digitale Gegenstücke physischer Objekte (Kritzinger et al. [23]).
2017	Digitale Zwillinge sind eine virtuelle Repräsentation eines Produktionssystems, welches verschiedene Simulationen durchführen kann. Weiters ermöglichen digitale Zwillinge mathematische Modelle und Echtzeit-Datenverarbeitung mit Hilfe erfasster Daten sowie angeschlossener Geräte (Negri et al. [24]).
2016	Ein digitaler Zwilling ist eine umfassende Beschreibung einer Komponente, eines Produktes oder eines Systems. Man erwartet von einem digitalen Zwilling, dass dieser mehr oder weniger über alle Daten verfügt, die in aktuellen sowie nachfolgenden Phasen hilfreich sein könnten (Boschert und Rosen [25]).
2016	Ein digitaler Zwilling ermöglicht die selbständige Simulation eines physischen Objekts zur Vorhersage zukünftiger Zustände des Systems (Gabor et al. [26]).
2015	Die Autoren beschreiben digitale Zwillinge als ein Modell, welches möglichst realitätsgetreu den aktuellen Zustand eines Prozesses sowie das Verhalten in Interaktion mit dem realen Umfeld widerspiegelt (Rosen et al. [27]).
2014	Zu den drei Hauptteilen digitaler Zwillinge gehören physische Produkte im realen Raum, virtuelle Produkte im virtuellen Raum sowie eine entsprechende Verbindung, welche jene beiden Räume miteinander interagieren lässt (Grievess [28]).
2012	Ein digitaler Zwilling ist ein ultrarealistisches Computermodell eines Flugzeuges, welches zur Überprüfung der Fähigkeit eines Flugzeuges zur Erfüllung dessen Missionsanforderung dient (Gockel et al. [29]).
2010	Ein digitaler Zwilling ist eine integrierte multi-physikalische, -skalische und probabilistische Simulation eines Fahrzeugs oder eines Systems. Hierbei sollen nicht nur die optimalsten physikalischen Modelle, sondern auch gesammelte Informationen sowie Daten von Sensoren und der Flotte genutzt werden. Somit kann das Leben der realen Repräsentanz wiedergespiegelt werden, damit ein oder mehrere wichtige und voneinander abhängige Fahrzeugsysteme berücksichtigt werden können (Shafto et al. [30]).

Tabelle 4.1.: Verschiedene Definitionen des digitalen Zwillings

4.2. Ursprung

Der erste Einsatz digitaler Zwillinge erfolgte noch vor der Einführung eines einheitlichen Begriffes und einer Vorstellung des Konzeptes [31]. Die ersten Zwillinge wurden 1970 im

4. Hintergrund: Digitaler Zwilling

Rahmen der Apollo-13-Mission von der NASA kreiert. Hierbei handelte es sich um Kopien von Weltraumkapseln, welche jenem Zweck dienten, aktuelle Flugbedingungen am Boden widerzuspiegeln. Jene Mission zum Mond war aufgrund eines defekten Sauerstofftanks zum Scheitern verurteilt. Weiters löste dieser Defekt eine Kette von unvorhergesehenen und nicht einmal in Betracht gezogenen Ereignissen aus. Anhand von Sensordaten sowie Simulationen konnten NASA-Ingenieure einen Weg finden, um schließlich die Astronauten sicher zurückzubringen [31]. Die Idee eines digitalen Zwillings reicht zurück bis in die 90er Jahre. Veröffentlicht wurde jene von David Gelernter, einem ehemaligen Professor an der Universität Yale, im Jahr 1991 in der Publikation *Mirror Worlds* [32]. Die Idee, ein Produkt während seines Lebenszykluses virtuell abzubilden und somit das Konzept eines digitalen Zwillings, kommt ursprünglich von Michael Grieves und wurde 2002 als Product Lifecycle Management (PLM) präsentiert [33]. Grieves machte sich bereits in den 90er Jahren über die Idee des digitalen Zwillings Gedanken, welche zu diesem Zeitpunkt noch über keinen verbreiteten Begriff verfügte. So nannte Grieves, der die Überzeugung vertritt, dass Industrie 4.0 nur mit digitalen Zwillingen möglich ist, die Idee eine Zeit lang *Doubleganger* [34]. Der heute geläufige Begriff *digitaler Zwilling* wurde 2010 von John Vickers, einem der leitenden Technikern der NASA, eingeführt [32]. Laut Krack et al. [35] bestehe der traditionelle, dreidimensionale digitale Zwilling, welcher 2003 von Grieves präsentiert wurde, aus einem physischen Gegenstand und dessen virtueller Repräsentanz sowie der Verbindung dieser beiden anhand von Daten und Informationen. Durch die Dimensionen Daten und Services wurde das klassische Modell erweitert, wodurch der fünfdimensionale Zwilling hervorging [35]. Laut dem Schweizer Portal Industrie2025 [36] haben sich in den letzten Jahren des Öfteren die Definitionen sowie die Wahrnehmung digitaler Zwillinge geändert. Während 2010 ein digitaler Zwilling als eine integrierte Simulation angesehen wurde, definierte man digitale Zwillinge phasenweise als originaltreue Modellierung (2014), reales Modell (2015), virtuelle Erweiterung (2016), digitale Kopie (2017), lebendes Modell (2018) und erweiterte virtuelle Instanz (2019). In der aktuellsten Phase geht es insbesondere um die Verfügbarkeit von Echtzeitdaten (2020) [36].

4.3. Treibertechnologien

Den neusten Fortschritten des Internets, dem IoT, Big Data, KI und anderen Informationstechnologien sind neue und wertvolle Möglichkeiten in unterschiedlichen Branchen zuzuschreiben. Vor allem im Zusammenhang mit digitalen Zwillingen finden solch fortschrittliche Technologien häufig ihren Einsatz [3]. Laut Landolsi et al. [37] kann mit Hilfe aktueller IoT-Technologien wie 5G, Datenanalyse, künstlicher Intelligenz, maschinellem Lernen (engl.: Machine Learning, ML) und Cloud-Computing zukünftig noch mit damit einhergehenden und revolutionären Fortschritten hinsichtlich digitaler Zwillinge gerechnet werden. So soll unter anderem verstanden werden, wie Produkte und Maschinen in Echtzeit miteinander interagieren. Weiters sollen Zukunftsvorhersagen schnell und genau ermittelt sowie potenzielle Fehler aufgedeckt werden können [37]. Laut Manca et al. [38] basieren digitale Zwillinge auf drei Säulen, bestehend aus Konnektivität, Digitalisierung und Intelligenz, welche anhand moderner Technologien ermöglicht werden.

4.3.1. Konnektivität

Sensoren

Digitale Zwillinge können mittels Sensoren Betriebs- und Umgebungsdaten physischer Prozesse der realen Welt erfassen. Gemäß Manca et al. [38] konnte durch der MEMS-Technologie (engl.: Micro-Electro-Mechanical-System) der Preis vieler Standardsensoren erheblich gesenkt werden, wovon jener Bereich besonders profitieren konnte [38].

IoT

Aktuell gewinnen IoT-Technologien aufgrund eines täglich zunehmenden Angebotes an unterstützten Anwendungen, Echtzeitüberwachung, Analysen und Entscheidungsunterstützungssysteme nach und nach an Bedeutung. Das industrielle IoT ist laut Kamath et al. [13] eine der wichtigsten Säulen hinsichtlich der Vision der Industrie 4.0 sowie eine Schlüsseltechnologie für digitale Zwillinge. Zur Integration eines IoT Gerätes existieren bereits zahlreiche IoT-Middleware-Lösungen bei welchen man die Wahl zwischen Open-Source und lizenzierter Software hat [13]. Hand in Hand mit der Entwicklung des IoT's entwickelt sich auch das Konzept intelligenter Objekte (engl.: Smart Objects). Solche können unter anderem Ereignisse in der physischen Welt sowie menschliche Aktivitäten wahrnehmen. Hierzu besteht die Umgebung aus intelligenten Objekten, Sensoren sowie einem Netzwerk aus intelligenten Systemen. Damit hinsichtlich einer engeren Interaktion mit der Umgebung und menschlichen Aktivitäten die Umgebungsintelligenz sowie Anpassungsfähigkeit und das Bewusstsein verbessert werden können, müssen digitale Zwillinge ähnlich wie ein intelligentes Objekt gestaltet werden [37].

5G Mobilfunk

Bis dato haben einige Länder mit der Einführung von 5G begonnen. Allerdings befindet sich die Technologie noch in der Angangsphase. Nichtsdestotrotz erhofft man sich durch 5G in naher Zukunft, dass bestehende wissenschaftliche sowie industrielle Herausforderungen gemeistert und somit das volle Potenzial dieser Technologie entfaltet werden kann. Hinsichtlich digitaler Zwillinge sind laut Nguyen et al. [39] 5G sowie 6G-Netze noch relativ neu. Die Nutzung digitaler Zwillinge, welche mit 5G ausgestattet sind, wären für einige Branchen mit Vorteilen verbunden und demnach von großer Bedeutung. So können unter anderem *Was-wäre-wenn-Analysen* für neue Produkte, Dienstleistungen oder Prozesse ermöglicht, neue Netzeinheiten (engl. Network Entities) erstellt und unter realistischen Bedingungen vor einer effektiven Umsetzung getestet werden. Weiters sollen reale Systeme kontinuierlich überwacht und Big-Data-Analysen sowie ML eingesetzt werden, um Probleme vorherzusagen, noch bevor solche in der realen Welt auftreten. Jenes soll dazu führen, dass der laufende Betrieb verbessert wird. Außerdem wird erwartet, dass sowohl die Implementierung sowie die Automatisierung von digitalen Zwillingen mit 5G profitieren können. So sollen Wartungsarbeiten reduziert sowie Produktionen optimiert werden können [39].

4. Hintergrund: Digitaler Zwilling

Big Data

Laut Manca et al. [38] handle es sich dabei um die Kombination von Betriebs- und Umweltdaten aus der realen Welt, welche mit Hilfe von Sensoren erfasst bzw. von einem Unternehmen bereitgestellt werden. Der Datenbestand, welcher sich durch IoT-Geräten anhäuft, ist riesig. Aus diesem Grund spielt bei der Entwicklung digitaler Zwillinge vor allem Big Data Analyse eine Schlüsselrolle. Durch solch eine Analyse sollen Probleme aus den gesammelten Daten extrahiert werden, um so Effizienz sowie Intelligenz in industrielle Prozesse zu bringen. Laut Rathore et al. [40] sind fortschrittliche Techniken, Architekturen, Frameworks, Tools und Algorithmen eine Grundvoraussetzung zur Verarbeitung solcher enormer Datenmengen. So eignen sich in der Regel für die Verarbeitung und Analyse von Big Data am besten Cloud-Computing Plattformen.

4.3.2. Digitalisierung

Simulationsmodellierung

Zur Abbildung der digitalen Seite eines Zwillings wird ein in Echtzeit laufendes digitales Modell der physischen Welt und Prozesse erstellt. Dieses Modell berücksichtigt explizites Wissen über den physischen Zwilling. Die Realisierung der Simulation hängt vom physischen Objekt ab. So könne ein mechanisches Ding mit physikalischen und mathematischen Techniken simuliert werden, während ein komplexeres System, beispielsweise ein Fließband, mit anderen Techniken wie der agentenbasierten Modellierung oder der diskreten Ereignissimulation modelliert werden [38].

Datengesteuerte Modellierung

Datengesteuerte Modelle (DDMs) dienen ebenso zur Abbildung der digitalen Seite eines Zwillings. Hierbei werden insbesondere mathematische sowie statistische Techniken zur Datenanalyse eingesetzt, um Beziehungen zwischen Ein- und Ausgaben eines Systems zu finden. Aufgrund großer Datenbestände, welche dank der schnellen Entwicklung fortschrittlicher Sensoren und Datenerfassungstechnologien zur Verfügung stehen, haben DDMs in den letzten Jahren an Bedeutung gewonnen [38].

4.3.3. Intelligenz

KI

Mit Hilfe von KI können Maschinen aus dem vergangenen Geschehen lernen, neue Eingaben einhalten sowie menschenähnliche Aufgaben ausführen. Insbesondere aufgrund immenser verfügbarer Datenbestände, der Fähigkeit zur Verarbeitung und dem Lernen anhand solcher Daten könne die Umgebung effektiv wahrgenommen, die Situation analysiert und die beste Entscheidung getroffen werden, um folglich das vorgegebene Ziel zu erreichen. Auf diese Weise können digitale Zwillinge eingesetzt werden, um unter Unsicherheit die beste Unternehmensleistung zu gewährleisten [38]. Laut Rathore et al. [40] könne ein intelligentes digitales Zwillingssystem nur mit Hilfe fortschrittlicher KI-Techniken, welche auf

die gesammelten Daten anwendbar sind, entwickelt werden. So erhofft man sich von dieser geschöpften Intelligenz, dass digitale Zwillingssysteme in der Lage sind, die besten Prozessstrategien zu erkennen sowie die besten Ressourcenzuweisungen, Sicherheitserkennungen sowie Fehlererkennungen vorherzusagen, zu optimieren und Entscheidungen dynamisch auf Grundlage von Sensordaten oder auch virtuellen Zwillingsdaten zu treffen [40].

Analytik

Laut Manca et al. [38] diene die Analytik dabei, aussagekräftige Muster in den Daten zu erkennen, zu interpretieren sowie zu kommunizieren, um schließlich effektive Entscheidungen treffen zu können. Hierzu verwende man diverse Analysetechniken, welche die Daten durch Algorithmen und Visualisierungsroutinen näher untersuchen. Insbesondere aufgrund der jüngsten Fortschritte im Bereich ML seien zahlreiche Anwendungen der Analytik entstanden, wodurch quantitative und qualitative Einblicke in Faktoren, welche das Unternehmen beeinflussen, ermöglicht werden [38].

Aktoren

Mit Hilfe von Aktoren werden angebrachte Aktionen in der realen Welt durchgeführt. Ausgelöst werden jene Aktionen von der KI des digitalen Zwillings, welche zuvor ermittelt, ob solche gerechtfertigt sind. Diese Ermittlung erfolgt auf der Grundlage operativer Entscheidungen oder Aktionen strategischer Natur [38].

4.4. Rolle digitaler Zwillinge in der Industrie 4.0

Digitale Zwillinge für die Industrie 4.0, auch bekannt als Verwaltungsschale, engl.: Administration Shell (AAS), speichern alle relevanten Industrie 4.0-Gegenstände. Solch ein Zwilling bilde den kompletten Lebenszyklus von Gegenständen ab und diene als Schnittstelle physischer Produkte zur digitalen und vernetzten Welt [41]. In einem Interview mit Dr. Michael Grieves [34], dem *Erfinder* des digitalen Zwillings, betonte dieser, dass Industrie 4.0 nur mit dem digitalen Zwilling möglich sei. Diese sollen in jener für Übersicht und Kontrolle sowie für Effizienz und Sicherheit sorgen. Laut Siemens [42] ebnen digitale Zwillinge stetig den Weg für die Industrie 4.0. Mit Hilfe jener können Produktionen optimiert sowie extrem hohe Prozessgenauigkeiten und so gut wie kein Ausschuss erreicht werden [42]. Dabei käme dem digitalen Zwilling aufgrund der engen Verbindung mit der vernetzten Produktion eine wichtige Überwachungsaufgabe zu [34]. Gemäß VDE [43], einer der größten Technologie-Organisationen Europas, sei es zukünftig möglich, dass jegliche Industrie-4.0-Komponenten uneingeschränkt von ihrem Produzenten und ohne weitere Beschränkungen in aufstrebenden Industrie-4.0-Netzwerke eingebunden werden können. Laut Kunz [44] seien digitale Zwillinge in Bezug auf die Industrie 4.0 deshalb so relevant, da sie den kompletten Entwicklungs-, Produktions- und Betriebszyklus eines Produktes oder Systems begleiten. So versuche man mit IoT-Sensoren ausgestatteten Produkten und deren angesammelten Maschinendaten Ausfallzeiten und Reparaturaufwände zu verringern oder ihre Produkte individueller zu gestalten. Die mit der Industrie

4. Hintergrund: Digitaler Zwilling

4.0 ausgelöste digitale Transformation führte dazu, dass seither neue Konzepte und notwendige Technologien entwickelt werden mussten. Hierbei verspricht man sich, insbesondere durch die Anwendung von digitalen Zwillingen, eine Performancesteigerung realer Systeme durch die Ansammlung, Analyse und Simulation von Daten [45]. Die Industrie 4.0 basiert laut Pires et al. [45] auf der Nutzung von cyber-physischen-Systemen (engl.: Cyber-Physical-Systems, CPS), Informations- und Kommunikationstechnologien (IKT) in Fertigungssystemen. Hierbei gelten vor allem das IoT sowie KI als entscheidende Erfolgsträger. Ziel der Industrie 4.0 ist es, dass die derzeitigen industriellen Systeme in flexiblere und anpassungsfähigere Systeme umgewandelt werden. Diese sind in der Lage zu lernen, sowie autonome Entscheidungen zu treffen [45]. Zu den wichtigsten Gestaltungsprinzipien zur Umsetzung der Industrie 4.0 zählen laut Pires et al. [45] Dezentralisierung, Interoperabilität, Virtualisierung, Echtzeitfähigkeit, Serviceorientierung und Modularität. Das hierbei erwähnte Konzept der Virtualisierung könne als Fähigkeit des CPS definiert werden. Mit Hilfe einer virtuellen Kopie eines physischen Systems, welches mit einem realen System verbunden wird, sollen Daten der physischen Welt gesammelt und überwacht werden können. Aus diesem Konzept habe sich im Laufe der Zeit eine neue Technologie entwickelt, die digitalen Zwillinge [45]. Jene dienen zum Verständnis als auch zur Optimierung der Interaktionen zwischen allen relevanten Material-, Energie- und Ressourcenflüssen in einem intelligenten Fertigungssystem (engl.: Smart Manufacturing System, SMS) [46].

4.4.1. Smart Manufacturing

Laut Lattanzi et al. [47] ging mit der Vision Industrie 4.0 ein neues Fertigungsmodell hervor, das sogenannte Smart Manufacturing. Dieses neue Konzept umfasst die Überwachung sowie die Steuerung realer Prozesse mit Hilfe digitaler Systeme. So sollen eine virtuelle Kopie der realen Welt geschaffen und dezentralisierte Entscheidungen getroffen werden können. Der Kern dieses neuen Konzepts sei dabei die enge cyber-physische-Integration, welche eine höhere Effizienz sowie Wettbewerbsfähigkeit in Fertigungsprozessen ermöglichen soll [47]. Laut Microsoft [48] verwende man digitale Zwillinge bei der Herstellung beispielsweise zur Optimierung von Abläufen, indem durch virtuelle Realität und Simulationen eine Fabrik und deren Prozesse gesteuert und folglich deren Auswirkungen auf das Produkt besser verstanden werden. Weiters dienen sie zur Überprüfung des Designs, indem die Produktion noch vor der Inbetriebnahme sowie vor Designänderungen getestet werden kann. Bei besonders komplexen Betrieben macht es Sinn, digitale Zwillinge auch zu Schulungszwecken einzusetzen. Hierbei können Mitarbeiter anhand von digitalen Nachbildungen beispielsweise für kritische Betriebssituationen, wie unerwartete Produktionsunterbrechungen, trainiert werden. Auf diese Weise soll die Sicherheit der Anlage verbessert werden, indem Mitarbeiter in solch einer Situation schnell und angemessen handeln [48].

4.4.2. Smart Factory

Eine Smart Factory, auch bekannt als Cyber-Physical-Production-System (CPPS), setzt insbesondere auf die skalierbare und modulare Struktur von Smart Manufacturing Anwendungen [49]. Laut Germanedge [50] könne eine Smart Factory ausschließlich anhand eines digitalen Zwillings und dessen Daten intelligent agieren und somit Auswertungen sowie Simulationen für diverse Produktionsszenarien durchführen. Der Einsatz industrieller IoT-Geräten (Industrial Internet of Things (IIoT)) erleichtert dabei laut Kamath et al. [13] die Echtzeit-Erfassung, -Verarbeitung sowie -Analyse großer Datenströme, welche von angebrachten Sensoren einer intelligenten Fabrik erzeugt wurden. Laut Kunz [44] werden künftig Unternehmen gesamte Fabriken in Form eines digitalen Zwillings abbilden. Solch ein Zwilling verfüge über Datenmodelle zur Beschreibung von Maschinen, Produktionsmittel, Gebäude, Versorgungseinrichtungen sowie ganzer Prozesse. Bei einer Smart Factory drehe sich alles um Betrieb und Optimierung in Echtzeit. Hierzu sei es erforderlich, dass die virtuellen Fabriken und Produktionssysteme ein exaktes Abbild der realen Fabrik widerspiegeln [44].

Deloitte [51] definiert eine Smart Factory als ein flexibles System, welches die Leistung in einem größeren Netzwerk eigenständig optimiert. Weiters sei eine Smart Factory in der Lage, in Echtzeit gesamte Produktionsprozesse autonom zu steuern sowie Anpassungen durch neue Bedingungen vorzunehmen und daraus zu lernen [51]. Laut Deloitte verfügen Smart Factories über fünf Hauptmerkmale:

- **Verbunden:** Zur Generierung der erforderlichen Daten müssen Prozesse und Materialien miteinander verbunden werden. Hierzu werden Anlagen mit intelligenten Sensoren ausgestattet [51].
- **Optimiert:** Laut Deloitte [51] ermögliche eine optimierte Smart Factory Abläufe mit minimalen manuellen Eingriffen und gleichzeitig hoher Zuverlässigkeit. So können solche den Ertrag, die Betriebszeit, Qualität, Kosten und Abfall positiv beeinflussen, indem Arbeitsabläufe automatisiert, Anlagen synchronisiert und die Nachverfolgbarkeit verbessert werden [51].
- **Transparent:** Anhand transparenter, erfasster Daten verfügen Smart Factories über transparente Netzwerke. Solche ermöglichen die Transparenz gesamter Anlagen und führen unter anderem zu präziseren Entscheidungen [51].
- **Proaktiv:** Ermöglicht zusätzlich zur korrekten Reaktion ebenso proaktives Handeln von Mitarbeitern oder Systeme noch vor dem Eintritt von Problemen oder Herausforderungen. Hierzu analysiere eine Smart Factory sowohl Echtzeit- als auch historische Daten [51].
- **Agil:** Darunter versteht man, dass Smart Factories in der Lage sind, anhand minimaler Eingriffe Änderungen umzusetzen. So könne unter anderem die Ausrüstung sowie der Materialfluss abhängig vom herzustellenden Produkt sowie Produktänderungen eigenständig konfiguriert und deren Auswirkungen in Echtzeit ermittelt werden [51].

4. Hintergrund: Digitaler Zwilling

Gründe für den Einsatz von Smart Factories sind zahlreiche Vorteile, wie Selbstkorrektur, erhöhte Gesamteffizienz, geringere Ausfallzeiten sowie Umrüstzeiten und optimierte Kapazitäten. Ebenso optimieren Smart Factories den Faktor Qualität, wodurch Mängel minimiert und somit Rückrufaktionen vermieden werden können. Optimierte Prozesse führen zu Verbesserungen der Kosteneffizienz und Reaktionszeit. Außerdem können durch erhöhte Betriebseffizienz, die Ausführung von wiederholenden und ermüdenden Tätigkeiten durch Maschinen sowie durch die Übertragung von Entscheidungen das Wohlbefinden und die Arbeitszufriedenheit der Mitarbeiter sowie die ökologische Nachhaltigkeit profitieren [51].

4.4.3. Unterschied zwischen digitalem Zwilling und Simulation

Laut IBM [32] verwenden sowohl digitale Zwillinge als auch Simulationen digitale Modelle, um Prozesse eines Systems nachzubilden. Durch die virtuelle Umgebung in Form eines digitalen Zwillings bieten sich dabei wesentlich mehr Analyse-Möglichkeiten. Im Wesentlichen bestehe der Unterschied im Simulationsumfang: eine Simulation untersuche demnach einen bestimmten Prozess, während ein digitaler Zwilling eine beliebige Anzahl nützlicher Simulationen durchführen könne, um folglich mehrere Prozesse zu untersuchen. Ein weiterer entscheidender Unterschied bestehe in der Zugriffsmöglichkeit auf Echtzeitdaten, welche normalerweise nur bei digitalen Zwillingen gegeben sei. Außerdem seien laut IBM [32] digitale Zwillinge auf einen zweiseitigen Informationsaustausch ausgelegt. So liefere ein Ausgangsobjekt mit Hilfe von Sensoren relevante Daten. Nachdem Erkenntnisse aus jenen Daten gewonnen wurden, werden diese an das ursprüngliche Ausgangsobjekt zurückgegeben. Mit Hilfe verfügbarer ständig aktualisierter Daten aus zahlreichen Bereichen in Kombination mit zusätzlicher Rechenleistung, welche die virtuelle Umgebung mit sich bringt, führen digitale Zwillinge des Öfteren zur Verbesserung von Produkten und Prozessen, da Probleme aus mehreren Blickwinkeln betrachtet werden [32].

Laut dem Forschungs- und Technologie-Institut TWI [52] können digitale Zwillinge die Genauigkeit analytischer Vorhersagemodelle erhöhen und ein besseres Verständnis für die Verwaltung und Überwachung von Produkten, Strategien und Verfahren ermöglichen. Weiters seien Simulationen statische Modelle, welche sich nicht weiterentwickeln oder verändern, es sei denn, zusätzliche Elemente werden eingefügt. Digitale Zwillinge hingegen sind dynamisch. Zwar beginnen digitale Zwillinge ähnlich wie Simulationen, jedoch sorgen Echtzeitdaten dafür, dass sich digitale Zwillinge ändern und entwickeln können, wodurch eine aktivere Simulation mit weiteren Informationen ermöglicht wird. Zusätzlich seien digitale Zwillinge spezifisch und real, Simulationen hingegen nur theoretisch. Während Simulationen abbilden, was mit einem Produkt geschehen könnte, zeigen digitale Zwillinge, was mit einem bestimmten Produkt in der realen Welt geschieht. Auf diese Weise könne ermittelt werden, ob das Produkt wie vorgesehen funktioniert. Weiters können auf Grundlage der tatsächlichen Nutzung Verbesserungen vorgenommen werden. Simulationen eignen sich insbesondere für die Produktentwicklung und ermöglichen das Testen verschiedener Szenarien anhand festgelegter Parameter. Der Anwendungsbereich digitaler Zwillinge reiche noch viel weiter. So werden sämtliche Phasen des Lebenszyklus eines Produktes von digitalen Zwillingen abgedeckt. Somit können digitale Zwillinge auch außerhalb des Designs eingesetzt werden und verhelfen zu Prozessverbesserungen und

zum Treffen umfassender Geschäftsentscheidungen [52].

4.5. Anwendung

Laut der offiziellen Webseite von IBM [32] werden digitale Zwillinge in unterschiedlichen Bereichen in großem Umfang eingesetzt. Hierzu zählen Energieerzeugungsanlagen, Bauwerke und deren Systeme, Dienstleistungen im Gesundheitswesen, die Automobilbranche, die Stadtplanung und Fertigungsprozesse. In Letzterer spiegeln digitale Zwillinge den gesamten Lebenszyklus eines Produktes wider. Dabei sind digitale Zwillinge in sämtlichen Fertigungsphasen allgegenwärtig geworden, was bedeutet, dass sie Produkte vom Entwurf bis hin zum fertigen Produkt und allen Schritten dazwischen begleiten. In den anderen aufgezählten Bereichen bieten digitale Zwillinge enormen Nutzen durch die Festlegung von Zeitplänen für regelmäßig erforderliche Wartungsarbeiten, verbesserte Planung und Erkenntnisse aus sensorgenerierten Daten. Weiters unterstützen digitale Zwillinge das Ziel, die Leistung herzustellender Produkte zu verbessern sowie die Produktion jener Produkte effizient zu steigern [32]. Herrlich [53] geht in seinem Artikel insbesondere auf den vielfältigen Einsatz digitaler Zwillinge in der Logistik ein. Mit diesem Konzept managen digitale Zwillinge gesamte Containerflotten, gestalten ihre Logistiksysteme und überwachen mit Hilfe von Sensoren Transporte. Zusätzlich könne durch IoT der Zustand vernetzter Maschinen, die Verfügbarkeit von Waren und Rohstoffen überwacht und erfasst werden. Köhler [18] erwähnt zusätzliche Anwendungsmöglichkeiten: in der vorausschauenden Wartung (engl.: Predictive Maintenance) zur Berechnung bevorstehender Ausfälle mittels einem Abgleich mit historischen Daten, im Qualitätsmanagement, in welchem digitale Zwillinge ein kontinuierliches Monitoring der Produktqualität bieten sowie zur Simulation in der Restrukturierung.

4.5.1. Anwendung in Fabriken

Digitale Zwillinge finden in verschiedensten Bereichen ihre Anwendung. In diesem Kapitel werden insbesondere Anwendungsfälle eines digitalen Zwillings in einer Fabrik vorgestellt, welche laut Köhler [54] zu Vorteilen einer Produktionsanlage über den gesamten Lebenszyklus der Fabrik führen können. Es folgt eine Auflistung von Anwendungsbeispielen, welche den vier verschiedenen Phasen des Lebenszyklus einer Fabrik gemäß Köhler [54] untergeordnet sind.

Phase 1: Planung, Bau

- **Anlagenplanung**

Laut Köhler [54] erhoffe man sich durch die Simulation einer Anlage Prozessabläufe zu optimieren sowie die frühzeitige Erkennung von Fehlern in Planung und Konstruktion, noch bevor jene Anlage in dieser Form realisiert wird. Hierzu können insbesondere Daten bereits existierender und vergleichbarer Anlagen zur Hilfe herangezogen werden [54].

4. Hintergrund: Digitaler Zwilling

- **Virtueller Fabrikzwilling**

Anhand solch einer virtuellen Abbildung könne man noch nicht gebaute Anlagen auf Aspekte wie *Mensch-Maschine-Zusammenarbeit* und Ergonomie prüfen. Weiters erschaffe man somit eine virtuelle Umgebung, in welcher die Anlage begehbar, erlebbar und interaktiv bedienbar sei [54].

Phase 2: Inbetriebnahme

- **Virtuelle Inbetriebnahme**

Wie die Bezeichnung schon vorwegnimmt, sollen neue Produktionsanlagen mit Hilfe des digitalen Zwillings vorab virtuell in Betrieb genommen werden. Diese helfe insbesondere die Gefahr von Fehlern zu reduzieren als auch Routinen und Checks vorzuverlegen [54].

- **Logistikplanung**

Laut Köhler [54] könne die gesamte Lieferkette (engl.: Supply Chain) rund um die Produktion durchgespielt werden. Somit könne mit Hilfe eines digitalen Zwillings beispielsweise die Produktionslogistik als komplexes System mit zahlreichen Parametern vor Produktionsstart simuliert werden [54]. Im Rahmen der Produktionslogistik wird der Transport geplant, gesteuert und durchgeführt sowie Rohmaterialien bis hin zu Fertigprodukten gelagert. Weiters umfasse diese ebenso damit zusammenhängende unterstützende Aktivitäten innerhalb des Produktionssystems eines Unternehmens [55].

Phase 3: Betrieb

- **Steuerung, Monitoring und Optimierung der Produktion**

Während des gesamten Betriebs der Anlage sollen sämtliche Daten der Simulation dem digitalen Zwilling zur Verfügung gestellt werden. Dies erlaube die Steuerung und das Monitoring des Prozesses sowie das Erkennen von Optimierungspotenzialen [54].

- **Qualitätssicherung**

Anhand digitaler Zwillinge könne die Einhaltung definierter Qualitätsparametern unterstützen sowie helfen, Ausschussraten zu minimieren [54].

Phase 4: Rekonfiguration

- **Modellwechsel**

Anhand des digitalen Zwillings sollen Produktionsumstellungen simuliert werden können. Somit lassen sich modifizierte Produktionsanlagen vorzeitig virtuell in Betrieb nehmen. Weiters könne man so Umrüstzeiten minimieren [54].

- **Customizing**

Darunter versteht man laut Köhler [54] die Simulation der Fertigung von Kleinserien bis hin zu individuellen Sonderanfertigungen.

4.5.2. Anwendung in Smart Cities

Digitale Zwillinge werden neben der Produktion unter anderem immer häufiger mit Smart Cities in Verbindung gebracht. Unter einer Smart City versteht man laut Botín-Sanabria et al. [56] einen strategischen Ansatz zur Integration von Daten und digitalen Technologien, welche die Nachhaltigkeit sowie das Wohlergehen der Bürger und die wirtschaftliche Entwicklung des städtischen Umfelds gewährleisten. Vorreiter in der Implementierung digitaler Zwillingstädte sind zum aktuellen Zeitpunkt China und Europa. Mit dem Einsatz solcher Zwillinge soll der Mensch in den Mittelpunkt gestellt werden, indem die Lebensqualität sowie verschiedene Dienstleistungen für Bürger verbessert werden sollen. Durch die digitale Nachbildung ganzer Stadtteile werden intelligente digitale Zwillingstädte erschaffen, in welchen städtische Netze einer Stadt, wie beispielsweise Strom- oder Verkehrsnetze, virtuell dargestellt werden. Auf diese Weise kann mit Hilfe von Geräten, welche in der Stadt verteilt sind und mit dem Internet verbunden sind, eine Überwachung ermöglicht werden. So könnten unter anderem neue Präventionsstrategien entwickelt oder die Auswirkungen der Einführung neuer Infrastrukturen im Voraus analysiert werden. Ein Beispiel aus der realen Welt wurde in Singapur implementiert. Hier wird ein digitaler Zwilling eingesetzt, um die Widerstandsfähigkeit der Stadt in unvorhersehbaren Szenarien zu erhöhen. Weiters können damit Umweltbedingungen in der Stadt überwacht sowie Notwegplanung in kritischen Situationen ermöglicht werden [45, 56].

4.6. Arten digitaler Zwillinge

Gleich wie bei der Definition eines digitalen Zwillings existieren zahlreiche unterschiedliche Arten, dafür aber keine einheitliche Unterteilung digitaler Zwillinge.

So unterteilt beispielsweise IBM [32] digitale Zwillinge in folgende Typen:

- **Komponenten- bzw. Teilzwillinge:** dabei handelt es sich um die Grundeinheit und kleinste Form eines digitalen Zwillings [32].
- **Asset-Zwillinge:** Bei einem Asset arbeiten zwei oder mehreren Komponenten zusammen. Dadurch kann näher untersucht werden, wie solche Komponenten miteinander interagieren [32].
- **System- bzw. Unit-Zwillinge:** Dabei handelt es sich um die nächste Vergrößerungsstufe. Systemzwillinge ermöglichen die Kombination verschiedener Assets, um folglich gesamte Systeme zu bilden. Solche Systeme können einen Einblick in Interaktionen von Anlagen gewähren und zu Leistungsverbesserungen führen [32].
- **Prozesszwillinge:** Indem gesamte Produktionsanlagen geschaffen werden, zeigen Prozesszwillinge wie verschiedene Systeme zusammenarbeiten. Dadurch lässt sich unter anderem ermitteln, ob die Systeme synchronisiert sind, ob mit höchster Effizienz gearbeitet wird oder ob sich Verzögerungen eines Systems auf andere auswirken. Mittels Prozesszwillingen soll die Gesamteffektivität positiv beeinflusst werden [32]. In einem Artikel von Microsoft [9] wird der *Process Digital Twin* vorgestellt. Demnach

4. Hintergrund: Digitaler Zwilling

umfassen solche Zwillinge gesamte Produktionsumgebungen und bieten virtuelle Nachbildungen des Geschehens in Fabrikhallen. Moderne Technologien sollen dazu verhelfen, gesamte Fertigungsprozesse zu optimieren. Außerdem unterstützen solche Zwillinge die Zusammenarbeit zwischen Menschen und Maschinen [9].

Krack et al. [35] unterscheiden drei Arten von digitalen Zwillingen:

- **Digitale Produktzwillinge:** Anhand eines solchen Zwillings können Produkte noch vor der realen Inbetriebnahme virtuell gefertigt werden. Eingesetzt werden jene Zwillinge in der Produktentwicklung und verfolgen das Ziel, Produkteigenschaften zu verbessern [35].
- **Digitale Produktionszwillinge:** Laut Krack et al. [35] verwende man solche Zwillinge beispielsweise bei der Planung von Fabriken. Sie dienen zur Unterstützung, Überwachung und Optimierung von Produktionsprozessen [35].
- **Digitale Performancezwillinge:** Bilden das *digitale Gedächtnis* eines Produktes oder einer Produktion anhand von Kennzahlen und Daten [35].

Chaplin et al. [57] unterteilen digitale Zwillinge in drei Klassen:

- **Digitale Überwachungszwillinge:** Hierbei handelt es sich um einen passiv agierenden Zwilling, welcher Daten von einem physischen System empfängt und diese zu einem einzigen Modell zusammenführt. Jene Klasse bezeichnet man auch als *digitalen Schatten* und wird vorwiegend zur Ermittlung von Warngrenzen und Schwellenwerten verwendet, welche auf Probleme hinweisen [57].
- **Interaktive digitale Zwillinge:** Solche Zwillinge werden typischerweise zur vollständigen oder teilweisen Kontrolle des physischen Gegenstücks eingesetzt [57].
- **Prädiktive digitale Zwillinge:** Diese Klasse trifft durch die Integration von Simulations- und Analysemethoden Vorhersagen basierend auf Prozessdaten des physischen Gegenstücks. Folglich sollen anhand jener Daten potenzielle Optimierungen von Prozessparametern ermittelt werden, welche bei Bedarf proaktiv vom physischen Gegenstück übernommen werden [57].

Die bekannten Geschäftsprozess-Tools Signavio und Adonis unterstützen bei der Erstellung eines DTOs, einer weiteren Art digitaler Zwillinge. Gartner [58] definiert jenen Zwilling als ein dynamisches Softwaremodell eines Unternehmens, welches operative sowie kontextbezogene Daten integriert. Somit soll ein Verständnis entwickelt werden, wie Unternehmen deren Geschäftsmodelle umsetzen. Weiters könne der Mehrwert, welcher sich für Kunden biete, sowie die Art und Weise wie auf Veränderungen reagiert und Ressourcen eingesetzt werden, ermittelt werden [58].

Digitale Prozesszwillinge

Laut Microsoft [9] erweitere ein Prozesszwillling den digitalen Zwillling eines Produktes über einzelne Maschinen hinaus und umfasse die gesamte Produktionsumgebung. Jener Zwillling setze hierbei auf Mixed Reality, KI und Hochleistungscomputer, um nicht nur die Maschinen, sondern den gesamten Fertigungsprozess zu optimieren. IBM [32] beschreibt Prozesszwillinge als eine Art digitaler Zwillinge, welche aufzeigen, wie Systeme zusammenarbeiten, um eine gesamte Produktionsanlage zu schaffen. Dabei wird untersucht, wie solche Systeme synchronisiert sind, um festzustellen, ob mit höchster Effizienz gearbeitet wird oder sich vielleicht Verzögerungen in einem System auf andere negativ auswirken. Ein weiterer Vorteil des Zwillings ist die gleichzeitige Nutzung virtueller und physischer Modelle anhand immersiver Hologramme sowie das ortsunabhängige, virtuelle Eintauchen ins Modell. Solche Hologramme empfangen laut Microsoft [9] Daten aus dem gesamten Prozess und stellen das gesamte System und Beziehungen zwischen seinen Komponenten dar. Anhand kognitiver Daten und KI einzelner Komponenten können Hersteller virtuelle Nachbildungen des Geschehens einer gesamten Fabrikhalle erstellen. Weiters profitieren jene von der optimierten Zusammenarbeit, die sich durch die Überbrückung der virtuellen und physischen Umgebung ergebe. Diese Umgebung unterstützt eine echte Maschine-zu-Maschine-zu-Mensch (M2M2P) Zusammenarbeit, was laut Microsoft zu einer 85%-igen Verringerung der Leerlaufzeit der Arbeiter bei der Zusammenarbeit mit Robotern führe. So können Techniker beispielsweise Änderungen durchführen, auch wenn die Maschinen in Betrieb sind oder sich jene/r an einem entfernten Standort bzw. vor Ort befindet [9]. In Deloitte's Ansatz für Industrie 4.0 [59] kombiniert eine digitale Fertigungsumgebung fortschrittliche Fertigungstechniken mit dem IoT. So solle anhand eines digitalen Zwillings eines Herstellungsprozesses sowohl ein vernetztes Produktionsunternehmen geschaffen, sowie Informationen kommuniziert, analysiert und genutzt werden, wodurch weitere intelligente Maßnahmen in der physischen Welt vorangetrieben werden sollen [59].

4.7. Elemente eines digitalen Zwillings

Chaplin et al. [57] nennen vier Elemente digitaler Zwillinge, wovon zwei in der Regel immer erforderlich sind:

- **Echtzeitverbindung:** Ein digitaler Zwillling **muss** mit einem System oder Produkt verbunden sein, welches über Sensoren verfügt, deren Daten mit dem digitalen Zwillling in Echtzeit geteilt werden können. [57].
- **Modell:** Ein weiteres **erforderliches** Element ist eine kontextschaffende virtuelle Nachbildung des physischen Gegenstücks in Form eines Modells sowie eine persistente Datenbank angesammelter Werte [57].
- **Analyse und Simulation:** Die Durchführung von Analysen und Simulationen dienen zum Verständnis physischer Gegenstände und sind häufig eingesetzte aber nicht zwingend notwendige Elemente digitaler Zwillinge [57].

4. Hintergrund: Digitaler Zwilling

- **Kontrolle:** Ein weiteres optionales Element eines digitalen Zwillings ist die Steuerung des physischen Gegenstücks [57]. Mit Hilfe von Aktoren können Parameter oder andere Elemente eines digitalen Zwillings nach Belieben umgestellt werden [57].

4.8. Architektur und Anforderungen

Anhand einer Architektur, welche die wichtigsten, erforderlichen technologischen Grundlagen umfasst, können die Initiativen des geplanten digitalen Zwillings problemlos umgesetzt werden. Gya et al. [60] gliedert die Architektur eines digitalen Zwillings in drei Schichten: einer Datenverwaltungsschicht, welche die Analyse von Daten ermöglicht, einer Modellbasierten Simulationsschicht zur Durchführung realistischer Simulationen sowie einer Geräte- und Softwareschicht, die es Mitarbeitern und Teams erlaubt, mit dem Zwilling zu interagieren [60]. Aufgrund signifikanter Unterschiede hinsichtlich eingesetzter Methoden, Protokollen und Treibertechnologien, variieren digitale Zwillings-Konzepte je nach Einsatzbereich. Nichtsdestotrotz hat die Architektur eines digitalen Zwillings, unabhängig vom Einsatzbereich, einen aus drei Elementen bestehenden Aufbau. Zu diesen drei Elementen gehören die physische, die virtuelle Welt sowie deren Verbindung. Angepasst an die entsprechenden Anforderungen werden in den einzelnen Elementen verschiedenste Komponenten integriert. In der Regel gehören zu den grundlegenden Elementen nicht nur Sensoren, die zur Abrufung von Informationen aus der echten Umgebung eingesetzt werden, sondern auch Datensicherheit, Datenverarbeitungsfunktionen und Kommunikationsschnittstellen wie unter anderem das Internet und Bluetooth. Ein ebenso wichtiger Bestandteil eines digitalen Zwillings ist die Datenvisualisierung für Nutzer [56]. In jener Architektur werden die physische Welt, bestehend aus Sensoren, Aktoren und Verarbeitungsmethoden sowie die digitale Welt, dazu gehören unter anderem der digitale Zwilling selbst, ML, Datenverarbeitungsfunktionen und Datenbanken, über ein Kommunikationselement miteinander verbunden. Hierfür können unter anderem verschiedene Protokolle und Schnittstellen wie beispielsweise das Internet oder Bluetooth eingesetzt werden. Die Umsetzung einer solchen Architektur ermöglicht dem Nutzer eine ständige Überwachung sowie Visualisierung. Kamath et al. [13] präsentieren eine digitale Zwillings-Architektur anhand eines Open Source Toolkits. Jenes umfasst eine Echtzeit-Datenerfassung, virtuelle Darstellung sowie Verwaltung, Echtzeit-Analyse und Visualisierung. Nach zahlreichen Leistungstests sowie -vergleichen war das Endresultat eine mittelschnelle, robuste, hochdynamische, skalierbare und quelloffene Architektur für IoT Smart Factory Lösungen. Zu den fünf eingesetzten Tools gehören *Eclipse Hono*, *Eclipse Ditto*, *Apache Kafka*, *InfluxDB* sowie *Grafana* [13]. Laut Gya et al. [60] müsse die Architektur eines digitalen Zwillings flexibel und in der Lage sein, fortschrittliche Datendienste wie Modellierung und Simulation, intuitive Visualisierung, Analytik und KI/ML sowie Event-Orchestrierung anzubieten. Somit sollen Vorhersage-/Präventivwarnungen und Echtzeitvorfälle effizient verwaltet werden können. Weiters sei insbesondere für die Skalierbarkeit des digitalen Zwillings ein Gleichgewicht zwischen Datengröße, Synchronisierung und Datenverarbeitung von entscheidender Bedeutung [60].

4.9. Herausforderungen

Digitale Zwillinge bieten zwar zahlreiche Möglichkeiten für Fabriken, jedoch haben Hersteller insbesondere bei der Umsetzung noch Schwierigkeiten. Laut einer von Microsoft [48] durchgeführten Befragung verursachen vor allem die Integration, die Entwicklungszeit der Erstellung des digitalen Zwillings und die Gesamtkomplexität des Systems die größten Herausforderungen. Weiters seien die Befragten der Meinung, dass es ihnen an erforderlichen Fähigkeiten mangelt und die Realisierung eines digitalen Zwillings mit relativ hohen Kosten verbunden sei. Die Befragten betrachten die Datenqualität, Datenmodellierung und Simulation nur als geringe Herausforderungen, weshalb für viele Unternehmen die Einstiegbarrieren digitaler Zwillinge nicht als größtes Hindernis der Realisierung angesehen werden [48]. Laut Balachandra und Prahallad [61] bestehe eine Herausforderung in der Ansammlung der Daten für den digitalen Zwilling. Anstatt nur möglichst viele verschiedene Datenströme heranzuziehen, sollen lediglich relevante Daten verwendet werden. Weiters erschweren fehlende oder unsichtbare Daten das Erlangen von Erkenntnissen wie beispielsweise Korrelationen. Neben technischen Herausforderungen sei es außerdem notwendig, dass die Mitarbeiter über eine ganzheitliche Denkweise verfügen, anhand welcher der digitale Zwilling betrieben und verwaltet werden kann. Zu weiteren Beispielen von Herausforderungen digitaler Zwillinge gehören Barrieren in Sensorsystemen und Betriebssystemen sowie seltene Klassenfehler [61]. Laut Chaplin et al. [57] gehören fehlende Standards und Interoperabilität zu weiteren Herausforderungen, wodurch die Einführung digitaler Zwillinge sowie die Kombination und Anpassung von Tools verschiedener Anbieter in der Regel sehr kostspielig oder eventuell gar nicht möglich sei. Eine weitere Hürde bestehe im Vertrauen in der Selbstoptimierung des Systems, welche aus Echtzeitdaten resultierende Ergebnisse aus Analysen und Simulationen nutzt, um folglich das System zu optimieren. Fehlendes Vertrauen könne dazu führen, dass der digitale Zwilling eine beratende statt der erwünschten integrierten Rolle einnehme. Auch die Echtzeit-Bereitstellung der zentralisierten Daten könne durch Faktoren wie der Datenmenge, der Entfernung der Datenquellen und der Datenübertragungsmethode den Versand von Rohdaten behindern. Zur Vorbeugung solcher Probleme könne anhand Datenvorbereitung die Datenübertragung erleichtert werden. Weiters seien insbesondere Unternehmen, welche IoT implementieren, des Öfteren Opfer von Cyberangriffen. Ein solcher Angriff auf einen digitalen Zwilling birgt das Risiko, dass das physische Gegenstück, sprich eine Maschine oder eine gesamte Anlage, stillgelegt oder Daten solcher gestohlen werden. So können unter anderem Software-Schutz sowie Datenverschlüsselungsmethoden eingesetzt werden, um Diebstähle zu verhindern. Eine weitere Herausforderung für Unternehmen mit unverständlichen oder nicht organisierten Daten bestehe im Implementierungsprozess des digitalen Zwillings [57].

4.10. Digitaler Zwilling Software

Tabelle 4.2 zeigt eine Gegenüberstellung einer Auswahl digitaler Zwillings-Software. Bei der Erstellung dieser konnte festgestellt werden, dass die meisten verfügbaren Programme

4. Hintergrund: Digitaler Zwilling

kostenpflichtig sind. Zudem zeigt die Gegenüberstellung, welche Programme PLC und 3D-Simulationen unterstützen. Weiters kann der primäre Zweck der Software abgelesen werden (DT = digitaler Zwilling Software, S = Simulationssoftware).

	NVIDIA OVX	NX / Kombi*	Eclipse Ditto	AWS IoT	Azure DT	FlexSim	Factory I/O
Hersteller	NVIDIA	Siemens	Eclipse	Amazon	Microsoft	FlexSim	Real Games
Open Source	✗	✗	✓	✗	✗	✗	✗
PLC Support	✗	✓	✓	✓	✓	✓	✓
3D-Simulation	✓	✓	✗	✓	✓	✓	✓
Kategorie	DT	DT	DT	DT	DT	S	S

* Hierbei handelt es sich um eine Kombination von Siemens Programmen.

Tabelle 4.2.: Gegenüberstellung digitaler Zwilling-Software

4.10.1. NVIDIA OVX

Mit *Omniverse Enterprise*, bereitgestellt in OVX, bietet NVIDIA eine kommerzielle, durchgängige 3D-Design-Kollaborations- und realitätsnahe Simulationsplattform, welche zur Entwicklung realitätsgetreuer 3D-Assets und -Umgebungen sowie umfangreicher, vernetzter digitaler Zwillinge eingesetzt werden kann. Anhand einer 30-tägigen Testversion und einer Version für Einzelpersonen und Community-Mitglieder, um die Betaversion des Software Development Kit (SDK) zu testen, besteht die Möglichkeit, die Software kostenfrei auszuprobieren [62].

4.10.2. Siemens Lösungen

Das Unternehmen Siemens bietet zahlreiche Services zur Erstellung digitaler Zwillinge an [63]. Zur Realisierung eines digitalen Produktionszwillings präsentiert Siemens [8] einen möglichen Ansatz, bei welchem eine Kombination hauseigener Programme angewendet werden. So dient der *Line Designer* zur Planung ganzer Produktionslinien und zur Auswahl der notwendigen Maschinen und Roboter für die Produktion. Folglich können mit Hilfe der *Tecnomatix Plant Simulation* und *Tecnomatix Process Simulate* Simulationen durchgeführt und Materialfluss-, Energiefluss- sowie Produktionsprozess-Optimierungen herbeigeführt werden. *SIMOVE* erlaubt es, eine modulare und flexiblere Produktion aufzubauen. Weiters können Standard-Automatisierungs- und Antriebskomponenten aber auch maßgeschneiderte Anwendungssoftware anhand einfacher Systemerweiterungen über offene Schnittstellen eingesetzt werden [8]. Laut Siemens biete deren kostenpflichtige Software *NX* Konstruktions-, Simulations- und Fertigungslösungen der nächsten Generation, anhand welcher digitale Zwillinge realisiert werden können. Es besteht die Möglichkeit, das Programm 30 Tage kostenlos zu testen [64].

4.10.3. Eclipse Ditto

Dabei handelt es sich um ein Open-Source-Framework und einer Technologie zur Erstellung digitaler Zwillinge von IoT-Geräten in der Cloud. Ditto ist horizontal skalierbar, zur Verwaltung von Millionen von Zwillingen konzipiert und erlaubt deren Integrierung in andere bestehende Backend-Systeme. Dadurch können Dinge wie jeder andere Webdienst über ihren digitalen Zwilling genutzt werden. Um mit Geräten zu interagieren, bietet die Technologie eine Hypertext Transfer Protocol (HTTP)- und WebSocket-Application Programming Interface (API) sowie Schnittstellen zur Nutzung von *Advanced Message Queuing Protocol* (AMQP), *Message Queuing Telemetry Transport* (MQTT) und Apache Kafka [65, 66, 67].

4.10.4. AWS IoT

AWS ermöglicht die Verbindung und Verwaltung von Milliarden Geräten anhand IoT-Services und -Lösungen. Dabei können IoT-Daten erfasst, gespeichert und analysiert werden. *TwinMaker* gehört zu den Analytik-Services, ermöglicht die Erstellung digitaler Zwillinge und dient folglich zur Optimierung von Produkten [68]. So ermöglicht jener Service unter anderem die digitale Abbildung realer Systeme wie Gebäude, Fabriken, Industrieanlagen oder Produktionslinien. Neben den Produkten sollen anhand diverser Tools ebenso die Leistung von Anlagen optimiert sowie eine ganzheitliche Sicht der Abläufe schneller und mit weniger Aufwand erstellt werden können. AWS Services, einschließlich AWS IoT TwinMaker, können 12 Monate kostenlos getestet werden [69].

4.10.5. Azure Digital Twins

Hierbei handelt es sich um ein IoT- Plattform-as-a-Service (PaaS) Angebot von Microsoft zur digitalen Darstellung von realen Dingen, Orten, Geschäftsprozessen und Menschen in Form von Zwillingengraphen. Laut Microsoft [70] können anhand der Plattform Einblicke gewonnen werden, welche die Produktentwicklung, Abläufe und Kosten optimieren sowie einzigartige Kundenerlebnisse schaffen können. Weiters könne eine Echtzeit-Ausführungsumgebung genutzt werden, um digitale Zwillinge in einer Echtzeit-Darstellung zum Leben zu erwecken. [70, 71]. Gleich wie Amazon, der direkte Konkurrent im Bereich Cloud-Computing, bietet auch Microsoft eine 12-monatige Testphase. Weiters erhält man ein Guthaben, welches innerhalb von 30 Tagen für nicht kostenfreie Services verwendet werden kann [72].

4.10.6. FlexSim

FlexSim ist eine leistungsfähige und benutzerfreundliche Simulationssoftware anhand welcher bestehende sowie geplante Systeme modelliert und verbessert werden können. Neben 3D-Simulationen können *Was-wäre-wenn*- sowie Modellanalysen durchgeführt und Optimierungen herbeigeführt werden [73]. Weiters kann die Software in regelmäßigen Abständen oder auch in Echtzeit Daten abrufen, weshalb sich diese ebenso zur Erstellung

4. Hintergrund: Digitaler Zwilling

digitaler Zwillinge eignet. Zusätzlich zur Testversion können Studentenlizenzen angefragt und kostenlos genutzt werden [74, 75].

4.10.7. Factory I/O

Factory I/O ist eine von Real Games entwickelte, kommerzielle Software, welche eine 3D-Fabrik-Simulation umfasst, anhand welcher Automatisierungstechnologien erlernt und ausprobiert werden können. Diese Simulationssoftware ermöglicht es seinen Nutzern auf eine möglichst einfache und intuitive Weise virtuelle Fabriken zu erstellen sowie eine Auswahl gängiger Industrieteile einzusetzen. Weiters bietet Factory I/O einen breiten Bestand an Szenarien typischer Industrieanwendungen an. Factory I/O wird überwiegend als PLC Trainingsplattform genutzt und kann gemeinsam mit Microcontroller wie Modbus, SoftPLC sowie einigen weiteren verwendet werden [76]. Laut Bécue et al. biete Factory I/O Simulationsaspekte digitaler Zwillinge, weshalb sich die Software zur Erstellung solcher eignet [63].

4.11. Digitale Zwillinge in der Zukunft

Laut Siemens [77], einem internationalen Unternehmen, welches unter anderem digitale Zwillinge Software entwickelt, werden digitale Zwillinge in naher Zukunft allgegenwärtig sein und eine noch viel wichtigere Rolle einnehmen als heute. Demzufolge sollen künftig Produkte mit deren Zwilling ausgeliefert werden, wodurch private als auch berufstätige Nutzer mit solchen interagieren können. Experten bei Siemens sind davon überzeugt, dass die Kombination von Simulation und künstlicher Intelligenz an Bedeutung gewinnen wird. Außerdem rechnet man in nahezu allen Bereichen mit digitalen Zwillingen und dass solche selbst zur Ware werden. Die Experten sind der Meinung, dass noch bis 2030 neue Geschäftsmodelle für digitale Zwillinge entstehen. Weiters sollen eigene Marktplätze für digitale Zwillinge geschaffen werden. Die Experten bei Siemens heben jedoch hervor, dass all jene Entwicklungen nur dann möglich seien, wenn die hierfür notwendigen Standards gesetzt werden. Aus diesem Grund spreche man der Standardisierung die Rolle des wichtigsten Motors der Entwicklung digitaler Zwillinge zu [77].

4.12. Zusammenfassung

Resümierend existieren zahlreiche unterschiedliche Definitionen sowie Arten digitaler Zwillinge. Aus der Recherche ging hervor, dass digitale Zwillinge zu den vielversprechendsten Technologien der heutigen Zeit zählen, insbesondere aufgrund ihres Einflusses zur Realisierung der Industrie 4.0. Das Konzept digitaler Zwillinge kann dank moderner Treibertechnologien wie dem IoT, Big Data und KI umgesetzt werden und in verschiedensten Bereichen Anwendung finden. In naher Zukunft müssen industriespezifische Standards für digitale Zwillinge erstellt werden, wodurch diese omnipräsent und eine noch wichtigere Rolle einnehmen werden.

5. Hintergrund: Workflow-Management-System

5.1. Geschäftsprozessmanagement

Der geläufige Begriff Geschäftsprozessmanagement (engl.: Business Process Management, BPM) ist laut Dumas et al. [5] sowohl eine Kunst, als auch eine Wissenschaft zur Überwachung der Arbeitsabläufe in einem Unternehmen. Dessen Anwendung solle einheitliche Ergebnisse sowie Verbesserungsmöglichkeiten herbeiführen. Hierbei lassen sich nun Parallelen zu digitalen Zwillingen ziehen, welche ebenso zur Realisierung von Optimierungsmöglichkeiten sowie zur Vermeidung von Fehlern und zur frühzeitigen Erkennung solcher Anwendung finden. So kann der Einsatz von BPM ebenso zu Verbesserungen hinsichtlich Kostensenkungen, Verkürzung der Durchführungszeiten sowie einer Verringerung der Fehlerquote führen. Dabei gehe es bei BPM nicht darum, wie einzelne Aktivitäten durchgeführt werden, sondern vielmehr wie eine ganze Kette von Ereignissen, Aktivitäten und Entscheidungen verwaltet werden sollen. Solche Ketten, welche laut Dumas et al. [5] letztlich einen Mehrwert für das Unternehmen generieren sollen, bezeichnet man als Prozesse.

5.2. Business Process Model and Notation (BPMN)

Hierbei handelt es sich um eine standardisiertes Geschäftsprozessmodell und eine Notation, deren aktuelle Version im Jahr 2010 von *The Object Management Group* (OMG) veröffentlicht wurde. Das Ziel besteht darin, dass alle Teilnehmer eines Geschäftsprozesses, unabhängig von ihrem Hintergrund, in der Lage sind, dessen Notation zu verstehen. Neben einer standardisierten Schnittstelle für das Geschäftsprozessdesign und der Prozessimplementierung ermöglicht BPMN außerdem die Visualisierung von Geschäftsprozessen anhand einer geschäftsorientierten Notation. Folglich diene BPMN als ein einfaches Kommunikationsmittel von Prozessinformationen für alle Stakeholder [78].

5.3. Workflow und WfMS

Unter einem Workflow versteht man einen mittels Softwaresystem vollständig oder zum Teil automatisierten Prozess. So wird hierbei garantiert, dass Informationen zwischen Teilnehmern zuverlässig weitergegeben sowie zeitliche und logische Abhängigkeiten, welche im Rahmen des zugrunde liegenden Prozessmodells verankert sind, eingehalten werden können. Dabei spricht man in der Literatur häufig von Process-Aware Information Systems

5. Hintergrund: Workflow-Management-System

(PAIS), eine eigene Art der Prozessoptimierung, welche das Wissen über verschiedene Aktivitäten eines Prozesses sowie deren Interaktion nutzt.

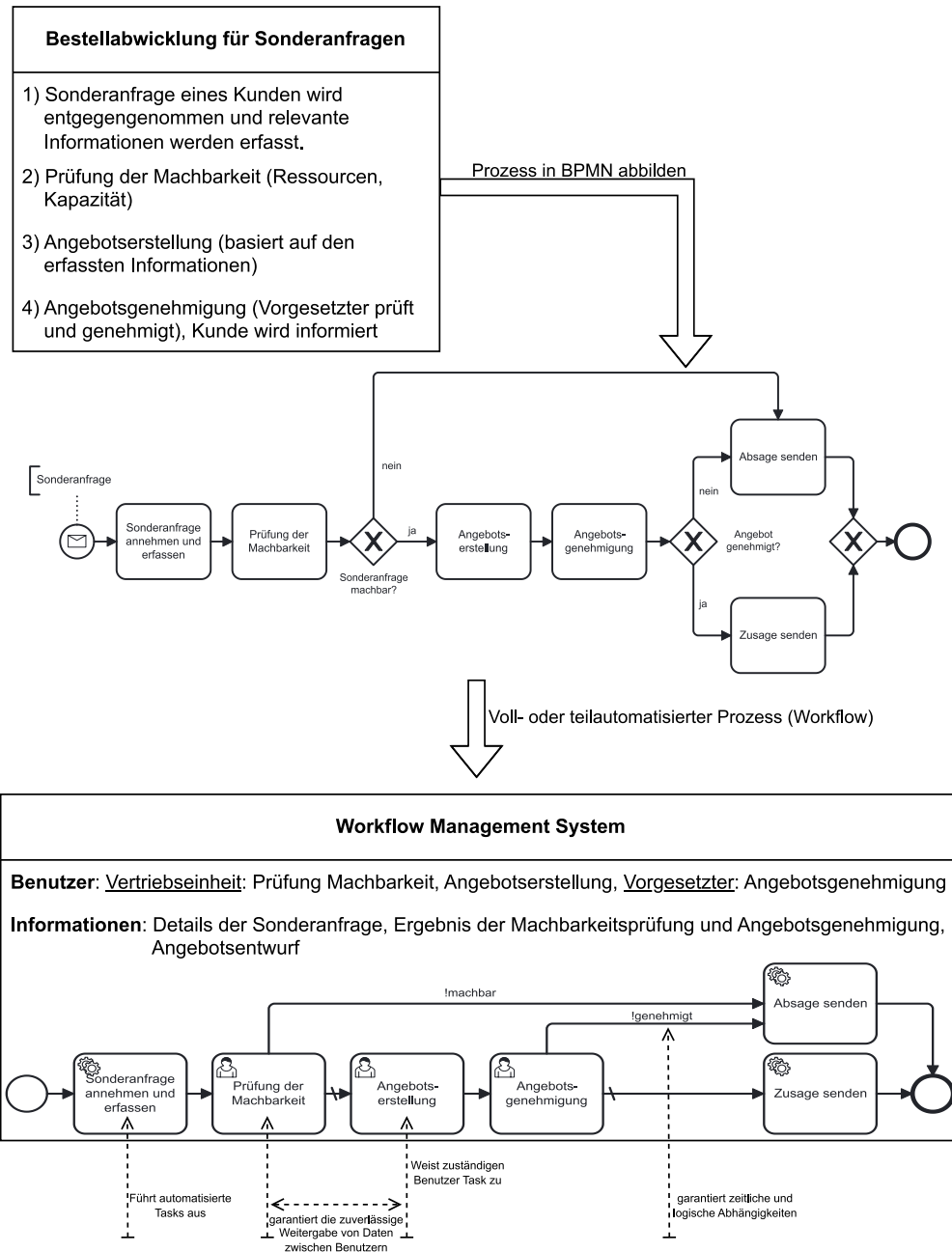


Abbildung 5.1.: Transformation eines Prozesses in ein Prozessmodell eines WfMS

Laut Dumas et al. [5] können konzeptionelle Prozessmodelle durch die Verwendung einer Sprache sowie Standards systematisch in ausführbare Prozessmodelle umgewandelt wer-

den. Diese können letztendlich von einem Business Process Management System (BPMS), auch WfMS genannt, interpretiert und automatisch ausgeführt werden. Anhand dieser Umwandlung soll schließlich das konzeptionelle Modell IT-orientierter werden [5].

Abbildung 5.1 zeigt eine textuelle Prozessbeschreibung sowie dessen abgebildeten Workflow sowie WfMS. Zur Modellierung des Prozesses wird BPMN verwendet, wodurch unter anderem die Auslösebedingung sowie sämtliche Tasks und Entscheidungen dargestellt werden können. Mit Hilfe eines WfMS kann schließlich jener Workflow ausgeführt und sichergestellt werden, dass der Kontrollfluss eingehalten wird. Außerdem wird gewährleistet, dass sämtliche Tasks auf die entsprechenden Benutzer aufgeteilt und die benötigten prozessrelevanten Informationen automatisch weitergegeben werden.

5.4. WfMS Architektur

Laut Dumas et al. [5] besteht die Architektur eines Workflow Systems aus folgenden Elementen:

- **Prozess-Modellierungs-Tool**
 - Ermöglicht Benutzern das Erstellen sowie Anpassen von Prozessmodellen.
 - Prozessmodelle können anhand von Daten ergänzt werden
 - Prozessmodelle können in einem Repository gespeichert und nach Belieben genutzt und abgerufen werden.
- **Prozess-Modell-Datenbank**
 - Erlaubt das Abrufen und Speichern von Prozessmodellen.
- **Verwaltungs- und Überwachungsinstrumente**
 - Überwachung der Ausführung von Arbeitsaufgaben
 - * Arbeitsaufgaben-Übersicht sowie Arbeitspensum je Teilnehmer
 - * Neuzuweisung von Arbeitsaufgaben bei speziellen Bedingungen wie Abwesenheit
 - * Behandlung von Ausnahmen
 - Überwachung von Geschäftsprozessen
 - * Überwachung der Performance
 - * Fortschritt einzelner Geschäftsprozessinstanzen
- **Ausführungsprotokoll (Execution Logs)**
 - Umfasst Aufzeichnungen prozessbezogener Ereignisse.
- **Arbeitslisten-Verwaltung (Worklist Handler)**

5. Hintergrund: Workflow-Management-System

- Weist Prozessteilnehmern Aufgaben zu und bietet jenen die Möglichkeit, den aktuellen Status solcher anzugeben, beispielsweise dass ein Task bearbeitet oder abgeschlossen wird.
- **Ausführungseinheit (Execution Engine)**
 - Ermöglicht die Erstellung ausführbarer Prozessinstanzen.
 - Um einen Prozess von Anfang bis Ende auszuführen, verteilt die Ausführungseinheit sämtliche Aufgaben auf die Prozessbeteiligten.
 - Ermöglicht ein ständiges Abrufen sowie Speichern von Daten, welche zur Ausführung des Prozesses sowie zum Delegieren automatisierter Aktivitäten an Software-Anwendungen erforderlich sind.
- **Externe Services**
 - Umfasst Dienste, welche bei der Ausführung des Geschäftsprozesses genutzt werden.

Die Ausführungseinheit ermöglicht unter anderem das Absenden von HTTP Anfragen und die Berücksichtigung der folglich gelieferten HTTP Antworten. Dies wird im Rahmen der Abschlussarbeit die Schnittstelle zwischen digitalen Zwillingen und dem Workflow bilden. Abbildung 5.2 zeigt die aufgezählten Elemente sowie deren Anordnung und Interaktionen in der Architektur von Workflows.

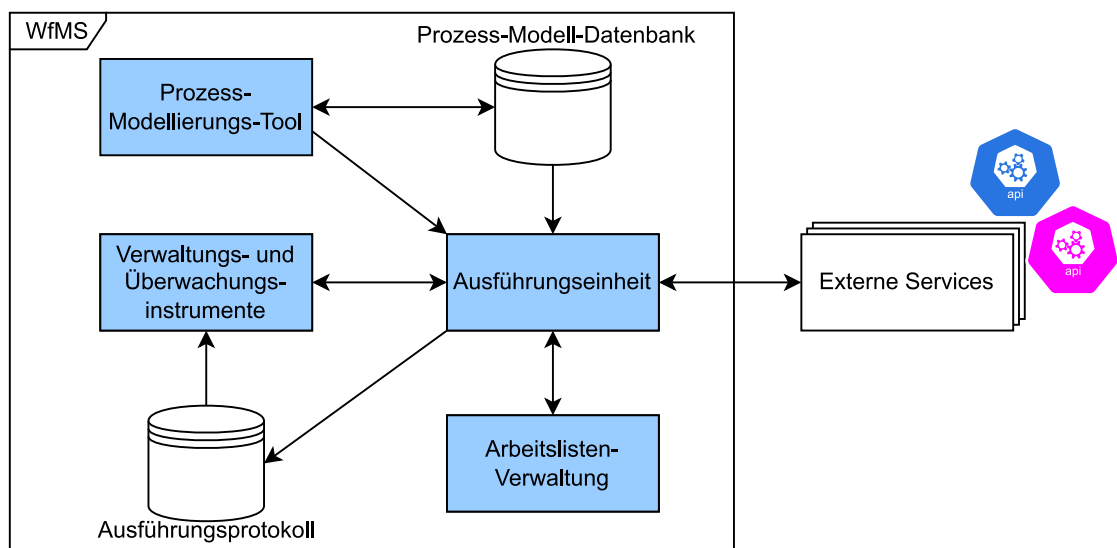


Abbildung 5.2.: Architektur von Workflows (Nachbildung [5])

5.5. WfMS Software

Die Suche nach WfMS Software hat ergeben, dass zahlreiche BPM und Modellierungstools verfügbar sind. Es folgen eine Gegenüberstellung einer Auswahl von WfMS Software sowie kurze Vorstellungen jener. Hierfür enthält Tabelle 5.1 Details hinsichtlich des Vorliegens von Funktionalitäten, einer REST-Schnittstelle und kostenlos zugänglicher Lizenzen oder Softwareversionen. Weiters beinhaltet die Gegenüberstellung Informationen zur Umgebung, in welcher die WfMS ausgeführt werden kann.

	ADONIS	Signavio	Bonita	AristaFlow	CPEE
Simulation	✓	✓	✓	✓	✓
Analyse	✓	✓	✓	✓	✓
REST	✓	✓	✓	✓	✓
Open Source	✗	✗	✓	✗	✓
Freie Lizenz	Community Edition	Academic Edition	Community Edition	Forschung & Bildung	LGPL V3.0
Cloud	✓	✓	✓	✗	✓
Lokale Ausf.	✗	✗	✓	✓	✓

Tabelle 5.1.: Gegenüberstellung WfMS-Software

5.5.1. AristaFlow BPM Suite [79]

Nachdem der *AristaFlow Server* gestartet wurde, können verschiedene Editoren zur Abbildung von Unternehmensstrukturen, Erweiterung von Aktivitäts-Repositories und Modellierung von Prozessen verwendet werden. AristaFlow stellt BPM-Funktionalitäten als REST-Services zur Verfügung, anhand welcher voll ausgestattete Workflow-Anwendungen implementiert werden können [79]. Neben der Option, AristaFlow entgeltlich zu erwerben, besteht die Möglichkeit, das Programm aus Forschungs- und Bildungszwecken kostenlos zu verwenden [80].

Signavio Process Manager [81]

Das kostenpflichtige Tool von SAP dient der Modellierung und Optimierung von Unternehmensprozessen. Signavio ermöglicht das Anlegen von Modellen, das Hinzufügen und Verbinden von Elementen sowie zahlreiche weitere Modellierungsfunktionalitäten. Weiters können Prozessläufe visualisiert und Prozesse Schritt-für-Schritt analysiert werden. Durch Prozesssimulationen sollen Erkenntnisse hinsichtlich Kosten, Durchlaufzeiten, Ressourcen und Engpässe gezogen und das Prozessbewusstsein erhöht werden. Zusätzlich bietet das BPMN-Simulationswerkzeug Simulationen für bestimmte Einzelfälle oder mehrerer Fälle durch konfigurierte Wahrscheinlichkeiten. Folglich besteht die Option, die daraus resultierenden Ergebnisse als Spreadsheet zu exportieren. Signavio unterstützt die Integration mit Anwendungen von Drittanbietern anhand einer REST-API [81].

5.5.2. ADONIS GPM-Suite [82]

Hierbei handelt es sich um ein kommerzielles sowie mehrsprachiges Werkzeug für Geschäftsprozessmanagement, welches zahlreiche Funktionalitäten für diverse Einsatzbereiche, wie beispielsweise der Geschäftsprozessoptimierung, bietet. Dabei sollen Prozesse, Rollen und weitere Unternehmensressourcen in einen Zusammenhang gebracht werden, um folglich einen digitalen Zwilling einer Organisation zu erstellen. Mit Hilfe von ADONIS können neue Modelle angelegt, Objekte hinzugefügt und verbunden sowie Modelle im grafischen Editor bearbeitet werden. Nach der Erfassung der Geschäftsprozessarchitektur in Form von Modellen können jene analysiert werden. Hierzu können anhand ADONIS gezielt nach bestimmten Modellen und Objekten gesucht, Dashboards eines Modells oder aus Reports angezeigt, unterschiedliche Modelle verglichen, Nutzenanalysen durchgeführt und Prozesse Schritt-für-Schritt analysiert (Prozessschrittanalyse) werden. Damit kann erkannt werden, wie Prozesse und einzelne Prozesspfade durchlaufen werden. Weiters ermöglicht ADONIS die Prozesssimulation, wodurch die Durchführung von Geschäftsprozessen simuliert werden kann. Dadurch lassen sich Durchlaufzeiten, Kosten und Ressourcen ermitteln und verschiedene Varianten von Prozessen testen, um einen potenziellen Verbesserungsbedarf zu ermitteln und quantitativ zu bewerten. Um anderen Anwendungen den Zugriff auf Daten von ADONIS zu ermöglichen, verfügt ADONIS über eine REST-Schnittstelle. Zum Testen von ADONIS kann die *Community Edition* verwendet werden, welche einen eingeschränkten Funktionalitätsumfang bietet [82].

5.5.3. Bonita [83]

Laut der offiziellen Webseite [83] ist Bonita eine erweiterbare Open-Source Plattform für die Automatisierung und Optimierung von Geschäftsprozessen. Zu den Zielen von Bonita gehören die Beschleunigung der Entwicklung, Inbetriebnahme und Wartung von Automatisierungsprojekten. Bonita lässt sich effizient in bestehende Informationssysteme integrieren und bietet umfassende Einblicke in die Prozessausführung von Unternehmen. Hierzu umfasst das WfMS die drei Hauptkomponenten *Bonita Studio*, *Bonita Runtime* und *Bonita Continuous Delivery*. Erstere verwendet man in der Entwicklungsphase eines Projektes. *Bonita Runtime* dient zur Ausführung in unterschiedlichen Laufzeitphasen und erlaubt Interaktionen über die REST- und Java APIs. Die letzte Komponente, *Bonita Continuous Delivery*, erleichtert die iterative Bereitstellung von Projekten [83].

5.5.4. Cloud Process Execution Engine (CPEE) [84]

CPEE ist eine kostenlose sowie modulare und serviceorientierte Ausführungseinheit, welche die Ausführung von Prozessmodellen ermöglicht. Dabei handelt es sich laut offizieller Dokumentation um ein Toolkit zur Erstellung kundenspezifischen Lösungen. Weiters könne die Execution Engine ohne großem Aufwand und Schwierigkeiten in Webseiten sowie Anwendungen eingebettet werden, da sich CPEE nur auf das HTTP-Protokoll stützt. CPEE unterstützt neben BPMN zahlreiche weitere Sprachen wie BPEL und YAWL. Zu den unterstützten Interaktionsprotokollen zählen SOAP, REST sowie XMPP. Weiters

bestehen Freiheiten hinsichtlich der Ausführungsgestaltung durch das Hinzufügen von REST-Diensten [84].

5.6. Workflow Dimensionen

Nach Aalst [85] bestehe ein Workflow aus mindestens drei Dimensionen:

- **Kontrollfluss-Dimension:** Kümmt sich um die Identifizierung sowie Weiterleitung von Tasks und garantiert die korrekte Reihenfolge jener während der Workflow-Ausführung [85].
- **Ressourcen-Dimension:** Hierbei handelt es sich entweder um menschliche Ressourcen wie Mitarbeiter oder nicht-menschliche Dinge wie beispielsweise Maschinen oder Software. Ressourcen werden in Rollen und Organisationseinheiten klassifiziert und zur Ausführung von Tasks eingesetzt [85].
- **Fall-Dimension:** Laut Aalst [85] befasse sich jene Dimension mit den individuellen Fällen, welche je nach Prozessdefinition von den entsprechenden Ressourcen ausgeführt werden.

5.7. Workflow Verifikation [86]

Die folgenden Informationen und Definitionen basieren auf der Theorie von Aalst [86] aus dem Jahr 1997, welche bereits umfassend untersucht wurden. Jene Definitionen sollen dazu dienen, ein gemeinsames Verständnis der verwendeten Begriffe in dieser Arbeit zu gewährleisten und eine Grundlage für weitere Diskussionen und Analysen zu schaffen.

5.7.1. Petri-Netze

Klassische Petri-Netze sind gerichtete zweistufige Graphen und wurden erstmals 1962 von Carl Adam Petri vorgestellt. Jene Graphen verfügen über zwei Knotentypen: Stellen und Transitionen. Die Verbindung zwischen den Knoten erfolgt durch gerichtete Kanten. Hierbei gilt, dass eine Verbindung zweier gleicher Knoten desselben Typs nicht erlaubt ist. Stellen sind passive Komponenten zur Abbildung des Zustandes oder einer Bedingung und werden durch Kreise dargestellt. Für Transitionen verwendet man hingegen Rechtecke. Jene bilden die aktiven Komponenten von Petri-Netzen und können den Zustand eines solchen ändern. Zur Modellierung dynamischer Systeme durch Petri-Netze, welche über eine statische Struktur verfügen, werden Markierungen eingesetzt, dessen Position nach bestimmten Regeln verändert werden kann. Markierungen werden als schwarze Kreise dargestellt und spiegeln den aktuellen Status eines Petri-Netzes anhand der aktuellen Verteilung solcher wider. Stellen können zu jedem Zeitpunkt über keine, eine oder mehrere solcher Markierungen verfügen. Verfügt eine Stelle über eine Markierung, so ist diese anwendbar. Insofern alle Eingangsstellen einer Transition anwendbar sind, kann diese gefeuert werden. Folglich wird jeder Eingangsstelle ein Token entnommen und schließlich

5. Hintergrund: Workflow-Management-System

einer zu jeden Ausgangsstelle hinzugefügt. Jene Bewegung von Markierungen gemäß den Auslöseregeln bezeichnet man als *Token Play* [85, 87].

Formale Definition von Petri-Netzen [86]:

Ein Petri-Netz ist ein sogenanntes *Triple* (P, T, F)

- P ist eine endliche Menge an Stellen
- T ist eine endliche Menge an Transitionen ($P \cap T = \emptyset$)
- $F \subseteq (P \times T) \cup (T \times P)$ ist eine Menge an Kanten (Transitionsrelation)
- Eine Stelle $p \in P$ ist eine Eingangsstelle einer Transition $t \in T$, nur dann, wenn es eine gerichtete Kante von p nach t gibt und wenn $(p, t) \in F$. Die Menge an Eingangsstellen für eine Transition t bezeichnet man als $\bullet t$.
- Eine Stelle $p \in P$ ist eine Ausgangsstelle einer Transition $t \in T$ nur dann, wenn es eine gerichtete Kante von t nach p und wenn $(t, p) \in F$. Die Menge an Ausgangsstellen einer Transition t bezeichnet man als $t \bullet$.
- $p \bullet$ und $\bullet p$ bezeichnen eine Menge an Transitionen, die p als Eingangs- bzw. Ausgangsstellen teilen.

Definition des Zustandes bzw. der Markierung von Petri-Netzen [86]:

- Ein Petri-Netz kann zu jedem Zeitpunkt über keine oder mehrere Markierungen verfügen: $M \in P \Rightarrow \mathbb{N}$.
- Verfügen beispielsweise die Stellen $p1$ und $p3$ über eine Markierung, Stelle $p2$ über zwei Markierungen und Stelle $p4$ über keine, so wird der Zustand wie folgt dargestellt: $1p1 + 2p2 + 1p3 + 0p4$ oder gekürzt $p1 + 2p2 + p3$

Bei einem Petri-Netz (P, T, F) und einem Anfangszustand M_1 gilt folgendes:

- $M_1 \xrightarrow{t} M_2$: Transition t ist anwendbar in Status M_1 und das Feuern von t in M_1 führt zu Zustand M_2 .
- $M_1 \rightarrow M_2$: es existiert eine Transition t , sodass $M_1 \xrightarrow{t} M_2$
- $M_1 \xrightarrow{\sigma} M_n$: die Feuersequenz $\sigma = t_1 t_2 t_3 \dots t_{n-1}$ welche vom Zustand M_1 zum Zustand M_n führt.

5.7.2. Workflow Netze (WF-Nets) [86]

Laut Aalst [87] seien high-level Petri-Netze ein geeignetes Tool zur Modellierung und Analyse von Workflows und WfMS. Hierzu modelliere man Tasks als *Transitionen*, Instanzen als *Markierungen* und kausale Abhängigkeiten als *Stellen* und *Kanten*. Wf-Nets sind eine Petri-Netz-Unterkategorie, beinhalten relevante Kontrollflussdimensionen eines Workflows und eignen sich insbesondere für die Darstellung, Validierung sowie Verifikation von Abläufen. Zudem erweitern diese klassische Petri-Netze um Konzepte zur Unterstützung von Geschäftsprozessen. Workflow-nets verfügen stets über zwei spezielle Stellen i und o , welche den Start (*Quellstelle*) bzw. das Ende (*Senkstelle*) eines Ablaufes darstellen. Ebenso ist von Stelle i nach Stelle o ein gerichteter Pfad für jede Transition erforderlich, d.h. alle Knoten müssen sich auf Pfaden vom Start- zum Zielort befinden, damit ein

Petri-Netz die Anforderungen eines WF-Nets erfüllt. Sämtliche Markierungen stehen bei WF-Nets für einzelne Prozessinstanzen.

Formale Definition von WF-Nets:

Ein Petri-Netz $PN = (P, T, F)$ ist nur dann ein WF-Net, wenn:

- PN über zwei spezielle Stellen verfügt: i und o . Stelle i ist eine Quellstelle: $\bullet i = \emptyset$. Stelle o ist eine Senkstelle: $o \bullet = \emptyset$. Grund dafür ist, dass für jede Instanz des WfMS, nachdem jene vollständig abgeschlossen wurden, diese wieder gelöscht werden. Demnach spezifiziert ein WF-Net den Lebenszyklus eines einzelnen Falles.
- Wenn eine Transition t^* zum Petri-Netz PN hinzugefügt wird, welche die Stelle o mit der Stelle i verbindet, sprich $\bullet t^* = \{o\}$ und $t^* \bullet = \{i\}$, dann ist das resultierende Petri-Netz stark verbunden. Dadurch werden Aufgaben und Bedingungen vermieden, welche keinen Beitrag zur Bearbeitung von Fällen leisten.

Jene genannten Anforderungen beziehen sich lediglich auf die Struktur eines Petri-Netzes und können statisch verifiziert werden. Eine weitere Voraussetzung lautet wie folgt: eine Instanz eines WF-Nets **muss** irgendwann beendet werden. Zum Zeitpunkt der Termination des Prozesses muss sich eine Markierung an der Stelle o befinden, während alle anderen Stellen leer sein müssen. Weiters sind keine *toten Aufgaben (Dead Tasks)* erlaubt, wodurch sichergestellt wird, dass sämtliche Aufgaben ausgeführt werden können, wenn man dem entsprechenden Weg durch das WF-Net folgt. Anhand dieser beiden zusätzlichen Voraussetzungen gewährleisten die Einhaltung der Soundness Eigenschaft.

Workflow-Net Eigenschaften [86]

Solidität (Soundness)

WF-Nets gelten gemäß derer Definition als strukturell solide.

- Für jeden Status M , welcher Status i erreichbar ist, existiert eine feuerbare Sequenz, welche von Status M zu Status o führt.

$$\forall_M(i \xrightarrow{*} M) \implies (M \xrightarrow{*} o)$$

- Status o ist der einzige erreichbare Status ausgehend von Status i mit mindestens einer Markierung in der Stelle o .

$$\forall_M(i \xrightarrow{*} M \wedge M \geq o) \implies (M = o)$$

- Es existieren keine toten Transitionen in (PN, i)

$$\forall_{t \in T} \exists_{M, M'} i \xrightarrow{*} M \xrightarrow{t} M'$$

Lebendigkeit (Live)

Für jede Transition existiert eine Feuersequenz ausgehend vom initialen Zustand i , welche jene anwendbar macht.

Beschränktheit (Bounded)

Hierfür müssen sämtliche Stellen für jeden Zustand eines Petri-Netzes beschränkt sein, damit sich keine Markierungen an einer Stelle gegen das Unendliche anhäufen.

5.8. Ausblick: Process Mining

Laut Sato et al. [88] handelt es sich beim Process Mining um Techniken, welche dem Verständnis und der Verbesserung von Prozessen in verschiedenen Anwendungsbereichen dienen sollen. Dabei sei jener Forschungsbereich derzeit im Wachstum, insbesondere aufgrund der erforderlichen Verfügbarkeit an Ereignisdaten sowie der Notwendigkeit, Geschäftsprozesse in sich schnell verändernden Umgebungen zu entwickeln und zu verbessern. Das Ziel von Process Mining sei es, nicht-triviale prozessbezogene Informationen aus Ereignisdaten zu extrahieren, welche von verfügbaren Informationssystemen aufgezeichnet werden [88]. So solle sich laut Rinderle-Ma und Mangler [11] ein qualitativ hochwertiger Bestand an Ereignisdaten ausgezeichnet zur Überwachung von Fertigungsprozessen während der Laufzeit eignen. Dabei unterstütze Process Mining unter anderem die Automatisierung, insofern passende Daten vorliegen. Die Prozess-Automatisierung könne hingegen integrierte und kontextualisierte Datensammlungen liefern, anhand welcher Process Mining Ergebnisse qualitativ verbessert werden können. Weiters sollen sich so neue Analyse-Methoden der Daten ergeben [11].

5.9. Zusammenfassung

In diesem Kapitel wurden theoretische Inhalte von Workflows kurz angeschnitten, indem zunächst der Begriff BPM erklärt und BPMN, die übliche Notation zur Visualisierung von Geschäftsprozessen, vorgestellt wurde. Im nächsten Schritt wurden Workflows und WfMS definiert und die WfMS-Architektur sowie Workflow-Dimensionen behandelt. Abschließend folgten eine kurze Einführung sogenannter WF-Netze und Process Mining sowie eine Gegenüberstellung und eine kurze Beschreibung gängiger WfMS-Technologien.

6. Gemeinsamer Einsatz digitaler Zwillinge und WfMS

In diesem Kapitel werden verschiedene Anwendungsgebiete der Literatur vorgestellt, in welchen sowohl digitale Zwillinge als auch WfMS von Bedeutung sind und/oder gemeinsam zum Einsatz kommen. Dabei liegt der Fokus insbesondere auf deren Interaktion und gegenseitigen Einfluss.

WfMS und digitale Zwillinge einer Organisation

Die beiden etablierten Geschäftsprozess-Tool-Anbieter SAP mit *Signavio* sowie BOC Group mit *ADONIS* ermöglichen die Erstellung digitaler Zwillinge. So ermögliche das kostenpflichtige *ADONIS Process Simulation* eine dynamische Analyse digitaler Zwillinge sowie das Ergreifen von Maßnahmen. *ADONIS Process Simulation* biete ein aus der Simulation resultierendes Dashboard, welche für weitere Analysen des digitalen Zwillings genutzt werden könne. Weiters unterstütze ADONIS umfassende Kostenanalysen, Kapazitätsplanung, Was-wäre-wenn-Analysen und vieles mehr [7]. Die führende Modellierungs- und Miningplattform Signavio bietet ebenso ein Tool zur Erstellung eines digitalen Zwillings an. Jener Zwilling solle gleichzeitig die Kommunikation erleichtern sowie Entscheidungsträger mit aktuellsten Daten versorgen. Wichtig anzumerken ist, dass es sich hierbei um einen DTO handle. Dies sei ein dynamisches Softwaremodell, welches eine virtuelle Visualisierung dessen biete, wie Prozesse ablaufen sollten und wie diese tatsächlich ablaufen [6]. Laut Signavio [89] helfen DTOs festzustellen, ob Geschäftsprozesse funktionieren und wie sich Prozessoptimierungen auswirken würden. Jene resultierende Zwillinge beinhalten jedoch beispielsweise keine virtuelle Nachbildungen des Geschehens einer gesamten Fabrikhalle, was unter anderem einen der wesentlichen Unterschiede zum digitalen Zwilling eines Prozesses nach der Definition von Microsoft [9] darstellt. Park und Van der Aalst [90] verwenden diverse Techniken zur Realisierung digitaler Zwillinge einer Organisation mit Hilfe von aktionsorientierten Process Mining. So werden unter anderem Überwachungstechniken eingesetzt und mittels Ergebnissen vorheriger Überwachungen Aktionen ausgelöst [90].

Modellgesteuerte PAISs

Drave et al. [91] präsentieren einen modellgesteuerten und generativen Softwareentwicklungsansatz zur Entwicklung von PAISs. Durch die Erweiterung von *MontiGem*, einem Generator der RWTH Universität Aachen für datenzentrierte Anwendungen, welcher

6. Gemeinsamer Einsatz digitaler Zwillinge und WfMS

zur Erstellung digitaler Zwilling Cockpits eingesetzt wird, können PAISs unter anderem aus UML-Klassendiagrammen und BPMN-Modellen generiert werden [91]. Digitale Zwilling Cockpits ermöglichen die Steuerung digitaler Zwillinge und vereinfachen die Modellierung, den Einsatz sowie die Überwachung solcher. Zudem fördern sie deren erfolgreiche Anwendung in der intelligenten Fertigung [92].

Modellgesteuerter digitaler Zwilling

Kirchhof et al. [93] zeigen eine modellgesteuerte Methode, welche die Verbindung zwischen dem abgebildeten System und dessen digitalen Zwilling erleichtert, da diese in der Regel mühsam und fehleranfällig ist. Weiters entkoppelt jener Ansatz die Entwicklung der CPS-Architektur sowie des digitalen Zwilling Informationssystems (DTIS). Letzteres umfasst graphische Benutzeroberflächen, wodurch ein CPS bequemer und einfacher verwaltet werden kann. Außerdem ist ein Digitales Zwilling Informationssystem (DTIS) verantwortlich für die Anzeige von Daten sowie der Interaktion zwischen Nutzern und dem CPS. Anhand verfügbarer CPS-Daten kann ein DTIS ebenso zur Überwachung solcher dienen [93].

Prozessbewusstes digitales Zwilling Cockpit (PADTC)

Bano et al. [94] präsentieren einen Prototyp für ein automatisiertes Krankenhaustransportsystem basierend auf ein PADTCs und einen Low-Code-Entwicklungsansatz. Unter Low-Code versteht man eine grafische Benutzeroberfläche mit vordefinierten Bausteinen, wodurch kein oder weniger eigener Code geschrieben werden muss [95]. Somit nehmen Domänenexperten eine aktive Rolle in der Prozessentwicklung des digitalen Zwilling ein. Mit Hilfe des Cockpits werden der digitale Zwilling gesteuert, Anweisungen an den physischen Zwilling erteilt und Aggregationen produzierter Daten verwendet, um ausschließlich relevante Informationen anzuzeigen. Weiters werde durch die Einbindung eines fixen Prozesses ein Kontext geschaffen. Hierzu werden Prozesse in Form von Prozessmodellen inkludiert, in welchen festgelegt wird, in welcher Reihenfolge und von wem oder was ein Task ausgeführt werden soll. Ein PADTC ermöglicht anhand einer Process Engine zudem die Handhabung von Prozessmodellen während der Laufzeit. Zur Erstellung des Prototypen wurde hierfür die Process Engine von Camunda verwendet [94, Video¹].

Die in Abbildung 6.1 dargestellte Visualisierung, basierend auf einer Darstellung in einem Video¹, verdeutlicht die Funktionalität eines digitalen Zwilling Cockpits. In diesem Kontext handelt es sich bei einem Cockpit um eine Benutzeroberfläche, die es ermöglicht, Daten abzurufen und zu visualisieren und mit Modellen des digitalen Zwilling zu interagieren. Zudem kann anhand des Cockpits das physische System teilweise überwacht sowie gesteuert werden, indem Befehle gesendet werden. Ein PADTC ist eine erweiterte Version

¹https://www.youtube.com/watch?v=6JEYAYTH62I&ab_channel=RWTH-Aachen-LehrstuhlSoftwareEngineering

des digitalen Zwillingscockpits, der zusätzliche Funktionen zur Handhabung expliziter Prozesse des physischen Objekts und seines Kontexts bietet [94].

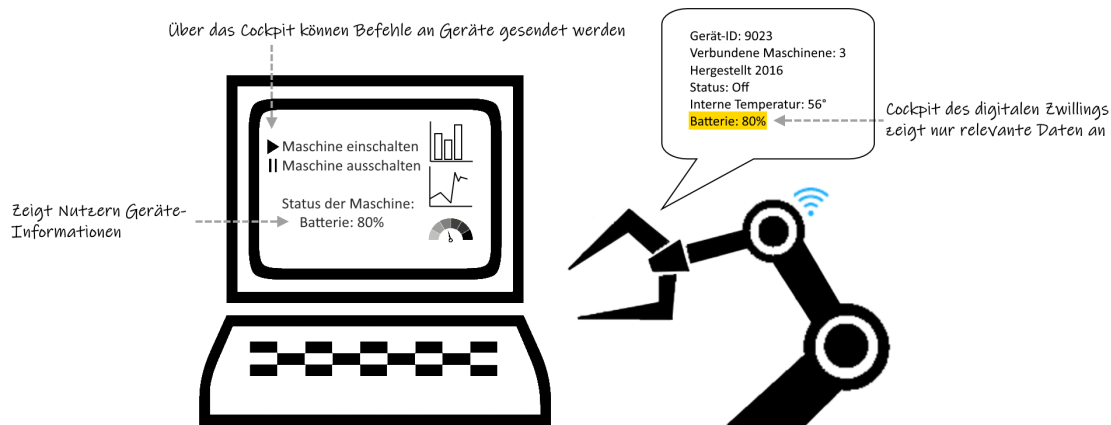


Abbildung 6.1.: Digitales Zwillingscockpit (Nachbildung¹)

Entscheidungsfindungsmodell eines digitalen Zwillings

Qamsane et al. [96] präsentieren einen Gesundheitsüberwachungsansatz und versuchen prädiktive Wartungslösungen in jenem zu ermöglichen, indem Open Process Automation (OPA) und digitale Zwillings-Technologien eingesetzt werden. Durch Leistungskennzahlen (KPI) der Überwachung sollen Instandhaltungsarbeiten zum richtigen Zeitpunkt durchgeführt sowie ungeplante Ausfallzeiten vermieden werden können. Dabei verwendet der digitale Zwilling ein Entscheidungsfindungsmodell, bestehend aus einer Zustandsmaschine (engl.: State Machine), welches zur Erkennung und Klassifizierung von Prozess- und/oder Steuerungsanomalien dient [96].

Rückgrat eines digitalen Zwillings

Rabah et al. [10] bezeichnen WfMS als Rückgrat sowie zentrale Komponente digitaler Zwillinge. Jene hohe Einschätzung lässt sich dadurch begründen, dass anhand WfMS unter anderem Workflows von Nutzern definiert und modelliert, Geschäftsprozess-Fristen festgelegt, Szenarien simuliert, Grenzen vorausgesehen und KPIs überwacht werden können. Weiters können digitale Funktionen und Workflows automatisiert und reibungslos ausgeführt werden [10].

7. Methodik

Im Rahmen dieser Arbeit werden Erkenntnisse und Daten sowohl von einer **Literatur-recherche** als auch einer eigenständig durchgeführten Implementierung erhoben. Die herangezogene Literatur beinhaltet moderne Lösungsansätze zur Implementierung eines digitalen Zwillings. Weitere miteinbezogene Quellen befassen sich sowohl mit verwandten Themengebieten digitaler Zwillinge als auch mit WfMS. Jene dienen zur Feststellung, inwiefern diese beiden Technologien in Kombination zum Einsatz kommen. Die übrigen Quellen dienen insbesondere zur Schaffung einer stabilen theoretischen Grundlage der Themengebiete. Zudem sollen diese zur Beantwortung der Forschungsfragen berücksichtigt werden. Weiters werden diverse Anwendungsfälle digitaler Zwillinge in einer Fabrik beleuchtet. Hierbei wird insbesondere darauf eingegangen, in welcher Lebenszyklus-Phase einer Fabrik solche sinnvoll eingesetzt werden können. Dies soll insbesondere dabei helfen, geeignete Anwendungsfälle zu definieren, welche in der Implementierung umgesetzt und schließlich in der Simulation durchgeführt werden können.

Das **Konzept** bzw. das Arbeitsvorhaben liegt darin, eine Fertigungslinie und deren Prozesse in einem 3D-Simulationsprogramm abzubilden. Jenes bildet gemeinsam mit einer Web-Anwendung, anhand welcher Echtzeitdaten abgerufen sowie Simulationen erstellt, verwaltet und durchgeführt werden können, den digitalen Zwilling, welcher schließlich von einem WfMS angesteuert und kontrolliert wird. Der Schwerpunkt liegt darauf, eine effektive Methode zu finden, um einzelne Objekte der Simulation wie Maschinen, Geräte, Sensoren, Pakete und Paletten mithilfe des WfMS zu platzieren und zu steuern. Besonders die Kommunikation zwischen den Geräten, Maschinen und Sensoren der Simulation mit dem digitalen Zwilling spielt hierbei eine wichtige Rolle. Das Ziel besteht darin, ein geeignetes Konzept zu definieren, welches insbesondere für IoT-Geräte geeignet ist.

Die aus dem Konzept resultierende **Implementierung** umfasst eine Fabriksimulation im Programm Factory I/O, welche mittels standardisierter PLC-Programmiersprachen automatisiert wird. Basierend auf dieser virtuellen Fertigungslinie soll mit Hilfe weiterer Technologien ein digitaler Zwilling erstellt werden, welcher über diverse Schnittstellen mit einem WfMS vernetzt und folglich von diesem gesteuert werden kann. Sämtliche Prozesse der virtuellen Fertigungslinie werden in BPMN-Notation abgebildet. Schließlich werden jene Prozesse vom WfMS ausgeführt, welches die Instanziierung mehrerer Prozesse ermöglicht sowie den Prozessablauf gemäß dem Prozessmodell regelt. Zudem können via HTTP-Anfragen externe Dienste genutzt werden. Durch die Verwendung vordefinierter Schnittstellen von Factory I/O, sprich der Web API und SDK, kann festgelegt werden, wie auf gewisse Ereignisse reagiert werden soll. Während das Simulationsprogramm ausgeführt wird, können *Tag*-Werte abgerufen werden, wodurch kontinuierlich Echtzeitdaten angezeigt und ein historischer Datenbestand angesammelt werden kann [97]. Hierfür werden bewährte Technologien und Protokolle eingesetzt, welche speziell für IoT-

7. Methodik

Geräte entwickelt worden sind. Auf diese Weise sollen Nutzer schließlich in der Lage sein, 3D-Simulationen anhand der Web-Anwendung und dem WfMS einzuleiten und zu überwachen. Des Weiteren können sie den Aufbau des zugrundeliegenden Prozesses, dessen Entscheidungsbedingungen und bei Bedarf auch die Services einzelner Tasks bearbeiten. Schließlich sollen zwei geeignete **Anwendungsfälle** herausarbeitet werden, bei welchen der digitale Zwilling der Fertigungslinie von einem WfMS kontrolliert werden kann. Der Fokus liegt hierbei auf der vollständigen Umsetzung von der Modellierung bis zur virtuellen Abbildung, Automatisierung und Steuerung. Hierzu werden die Anwendungsfälle in BPMN-Notation modelliert. Anschließend werden wichtige Prozess-Eigenschaften kontrolliert und der Aufbau der 3D-Simulation erläutert. Die beiden Anwendungsfälle, welche auf einer einzigen Fertigungslinie einer virtuellen Fabrik mit Hilfe von Factory I/O simuliert werden können, lauten wie folgt:

Anwendungsfall 1: Produktion von Möbel:

Die Fertigungslinie ermöglicht die Produktion zweier unterschiedlicher Produkte. Anhand zusätzlicher individueller Produktionsangaben sollen verschiedene Produktionen getestet werden können. Mit Hilfe des WfMS und Sensoren sowie Radio-Frequency Identification (RFID)-Leser soll hierfür der korrekte Materialfluss gewährleistet werden.

Anwendungsfall 2: Verpackung und Versand:

Nachdem die Produktion abgeschlossen wurde, landen sämtliche Produkte auf der Verpackungsstraße. Hierbei wird der entsprechende Karton zur Verpackung, abhängig vom Produkttyp sowie Bestelldetails, zugeordnet. Anschließend erfolgt die Versendung des Pakets, basierend auf dessen Größe, Gewicht und Bestelldetails.

8. Konzept

Im folgenden Kapitel wird das Konzept zur Steuerung eines digitalen Zwillings anhand eines WfMS beschrieben. Dabei wird erläutert, welche Vorteile sich aus diesem Konzept ergeben und wie die Interaktion zwischen WfMS und digitalem Zwilling funktioniert. Weiters werden die vorgesehenen Möglichkeiten vorgestellt, welche sich aus der Anwendung der Web-Anwendung des digitalen Zwillings ergeben. Diese wird im Folgenden als Digital Twin Simulation Application (TWIN) bezeichnet und soll eine Schnittstelle zum digitalen Zwilling bieten.

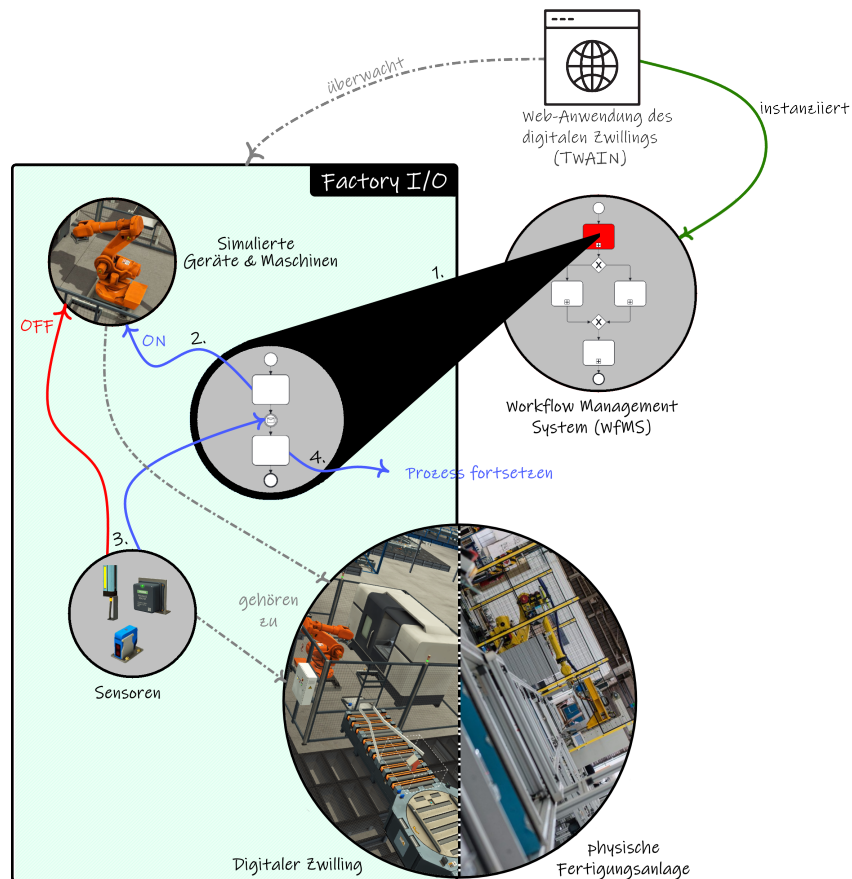


Abbildung 8.1.: Konzept zur Steuerung eines digitalen Zwillings anhand eines WfMS

8.1. Konzeptuelle Steuerung eines digitalen Zwillings anhand eines WfMS

Das Konzept sieht vor, dass der Kontrollfluss sowie sämtliche Entscheidungen des digitalen Zwillings vom WfMS kontrolliert werden. Dadurch werden Tasks von der Workflow-Engine gestartet und nach dessen Abschluss von jener abgeschlossen. Folglich kann dadurch der Prozess-Fortschritt der Simulation in der Workflow Engine angezeigt werden. Die aus der Simulation resultierenden Daten werden gespeichert und als Dashboard aufbereitet. Außerdem besteht die Möglichkeit, Echtzeitanalysen durchzuführen. Sämtliche Ist-Zustände und Werte von Geräten etc. werden in der TWAIN angezeigt, um zu sehen, für welche Bestellungen bzw. Simulation einzelne Maschinen oder Geräte gerade benötigt werden. Durch die Steuerung des digitalen Zwillings mittels WfMS bietet sich insbesondere für Nicht-Spezialisten ein großer Mehrwert, da der zugrunde liegende BPMN-Prozess für alle Parteien verständlich und abänderbar ist. Dadurch können beispielsweise die Bedingungen für Entscheidungen ganz einfach neu definiert oder abgeändert werden, ohne dass ein tiefgründiges Verständnis bzgl. PLC oder Automatisierung benötigt wird.

Abbildung 8.1 zeigt das Konzept der Interaktion zwischen WfMS und digitalem Zwilling, bestehend aus vier erforderlichen Schritten. Für die Umsetzung des Konzepts sind Sensoren, Geräte, Maschinen und RFID-Objekte notwendig, welche allesamt Bestandteile des abgebildeten digitalen Zwillings einer Fertigungslinie darstellen.

1. **Schritt 1:** Nachdem ein Nutzer eine Simulation über die TWAIN instanziiert hat, leitet das WfMS den Prozess ein und startet einen Task. Dabei können via der Anwendung durchgehend Echtzeitwerte sowie Zustände von Maschinen und Geräten überwacht werden. Jeder Task besteht wiederum aus zwei Sub-Tasks und einem Zwischenereignis. Das WfMS pausiert den Workflow und wartet auf ein Zwischenereignis.
2. **Schritt 2:** Erforderliche Geräte bzw. Maschinen der virtuellen Fertigungslinie in der 3D-Simulationssoftware Factory I/O werden eingeschaltet.
3. **Schritt 3:** Sensoren schalten Geräte bzw. Maschinen wiederum aus und geben ein Zwischenereignis zurück.
4. **Schritt 4:** Sobald das Zwischenereignis eintritt, wird der Task abgeschlossen und der Workflow fortgesetzt.

8.2. Entwurf der Digital Twin Simulation Application

Die geplante Web-Anwendung für den digitalen Zwilling bietet eine umfassende und intuitive Schnittstelle, die es Benutzern ermöglicht, einfach mit dem digitalen Zwilling sowie WfMS zu interagieren. Eine untergeordnete Seite zur Definition und Erstellung von Simulationen ist ebenfalls enthalten.

8.2. Entwurf der Digital Twin Simulation Application

Die Anwendung ermöglicht die Ausführung von Simulationen und die Anzeige des Fortschritts sowie Prozessdaten. Benutzer haben die Möglichkeit, den aktuellen Zustand von Maschinen, Geräten und Sensoren in Echtzeit einzusehen, einschließlich der Task Übersicht von Maschinen und Geräten. Hierdurch können Nutzer beispielsweise erkennen, welche Bestellungen bzw. Simulationen derzeit von einer Maschine bearbeitet werden oder sich in dessen Warteschlange befinden. Darüber hinaus bietet die Anwendung ein Dashboard, über das Benutzer Einblicke erhalten können, indem die für sie relevanten Informationen und Statistiken auf einen Blick zu sehen sind.

Abbildung 8.2 zeigt das Anwendungsfalldiagramm der TWAIN. Diese bietet den Benutzer fünf verschiedene Anwendungsmöglichkeiten.

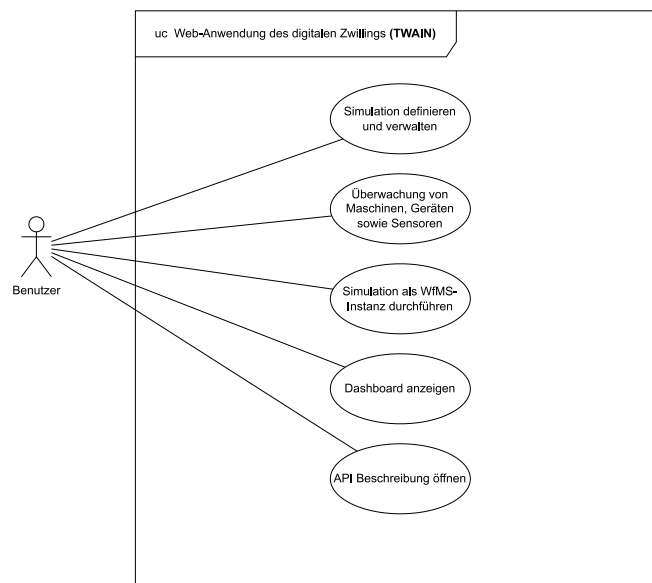


Abbildung 8.2.: Anwendungsfalldiagramm der TWAIN

1. **Simulation definieren und verwalten:** Benutzer können über ein simples Formular eine neue Simulation definieren, herunterladen und löschen. Dadurch können neue, für den Prozessablauf relevante Bedingungen, festgelegt werden, um während der Durchführung dementsprechende Entscheidungen treffen zu können.
2. **Simulation als WfMS-Instanz durchführen:** Ermöglicht Benutzern das Starten, Stoppen sowie die vollständige Durchführung und Kontrolle der Simulation anhand des WfMS.
3. **Überwachung von Maschinen, Geräten sowie Sensoren:** Benutzer können stets via der Simulationsausführungs-API Echtzeitzustände sowie -werte von Sensoren, Geräten und Maschinen sowie deren Aufträge abrufen.
4. **Dashboard anzeigen:** Anhand des Dashboards können Benutzer Daten aus ei-

8. Konzept

ner Echtzeitdatenbank in aufbereiteter Form abrufen und analysieren. Um ein Dashboard zu bearbeiten oder ein neues zu erstellen, können über die Grafana-Benutzeroberfläche (Port 3000) solche als JavaScript Object Notation (JSON)-File gespeichert und im *Provisioning*-Verzeichnis (siehe GitHub-Repository Ordner *grafana*) unter *dashboards* diese hinterlegt werden.

5. **API Beschreibung öffnen:** Erlaubt es Nutzern die Services der API zu testen sowie die Beschreibungen jener abzurufen.

8.3. Weitere Aspekte

Zuverlässigkeit und Skalierbarkeit

Jene beiden Aspekte betreffen die Kommunikation zwischen Sensoren, Maschinen und Geräten der Simulation mit dem digitalen Zwilling. Hierzu sollen für IoT-Geräte geeignete Technologien wie ein standardisiertes Kommunikationsprotokoll eingesetzt werden, um eine lose Kopplung sowie Skalierbarkeit in der Kommunikation zu ermöglichen. Außerdem sollen diese noch zahlreiche weitere Vorteile wie eine effiziente Datenübertragung, hohe Durchsatzraten und Robustheit mit sich bringen, was im Wesentlichen zu einer verbesserten Gesamtleistung der Implementierung führt.

Benutzerfreundlichkeit und Allgemeinverständlichkeit

In der Implementierung sollen sowohl Benutzerfreundlichkeit als auch Allgemeinverständlichkeit insbesondere durch das Vorliegen eines Dashboards und der Verwendung von BPMN-Modellen sowie der darauf basierten Entscheidungsfindung durch das WfMS gefördert werden. Dies erleichtert die Nutzung der Simulationsumgebung für verschiedene Benutzergruppen und ermöglicht eine effiziente Überwachung, Steuerung und Anpassung von Prozessen. Da die Prozesse in der Simulationsumgebung auf BPMN-Modellen basieren, ist die Allgemeinverständlichkeit gewährleistet. BPMN-Modelle sind darauf ausgelegt, nicht nur von Spezialisten, sondern auch von anderen Fachleuten verstanden zu werden. Dies ermöglicht eine breitere Nutzung und Zusammenarbeit in der Simulationsumgebung. Statt nur den Zugriff auf Rohdaten zu gewähren, wird ein Dashboard aufgesetzt, das einen übersichtlichen und leicht verständlichen Überblick über relevante Informationen bietet. Außerdem sollen Benutzer aktuelle Werte von Geräten oder anderen Sensoren in der 3D-Simulation in einer Web-Anwendung abrufen können. Dies soll Benutzern eine schnelle und einfache Überwachung der Simulationsumgebung ermöglichen, um folglich Entscheidungen auf Grundlage von aktuellen Daten treffen zu können. Das WfMS trifft Entscheidungen im Prozessablauf basierend auf den definierten Regeln und Bedingungen. Wenn beispielsweise eine Bedingung geändert wird, kann die Simulation mit den neuen Spezifikationen durchgeführt werden, indem die Anpassungen im Prozess vorgenommen werden. Dies ermöglicht eine flexible und benutzerfreundliche Anpassung von Prozessen in der Simulationsumgebung, ohne dass umfassende technische Kenntnisse erforderlich sind.

9. Technische Implementierung

Dieses Kapitel widmet sich der umfassenden Beschreibung der technischen Implementierung, beginnend mit der Darstellung einer abstrakten Architektur. Im Anschluss wird die konkrete Architektur präsentiert, welche zur Umsetzung des Prototyps eingesetzt wird. Dabei werden sowohl eigenentwickelten Lösungen eingeführt sowie verwendete Drittanbieter-Softwarekomponenten erläutert. Das Ziel dieses Kapitels besteht darin, eine detaillierte Darstellung der Funktionsweise, der einzelnen Bestandteile sowie der Gesamtimplementierung zu bieten, um Leserinnen und Lesern die Möglichkeit zu geben, das gezeigte Beispiel anhand des bereitgestellten Codes erfolgreich zu reproduzieren und vollständig nachvollziehen zu können.

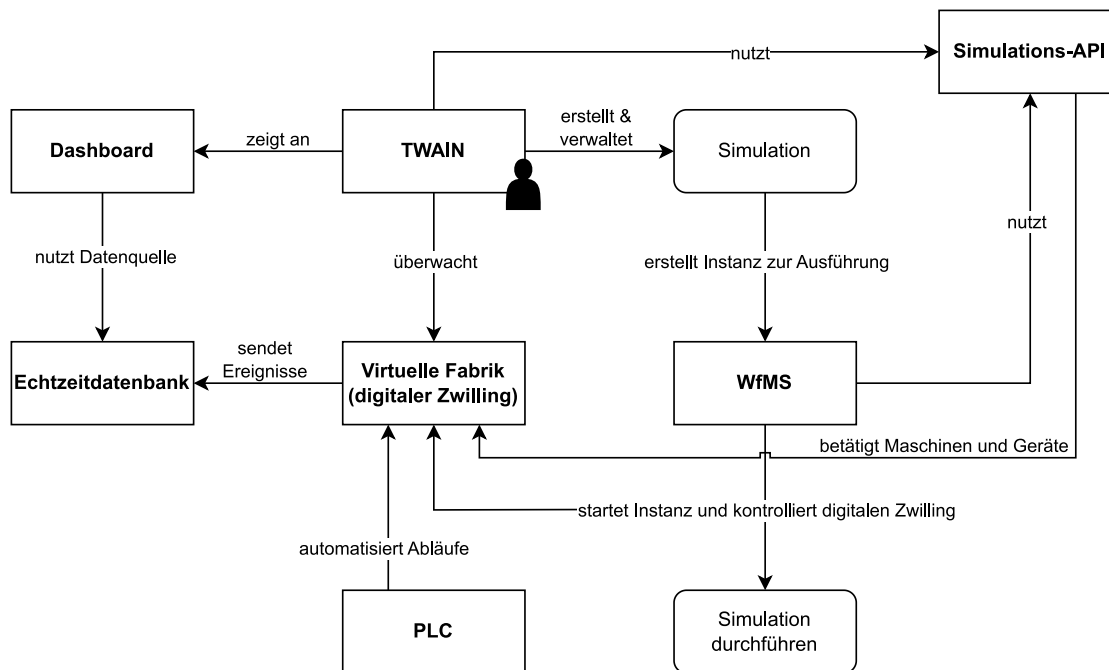


Abbildung 9.1.: Abstrakte Darstellung der Softwarearchitektur

Abbildung 9.1 zeigt eine abstrahierte Architektur zur Steuerung eines digitalen Zwillings einer virtuellen Fertigungslinie anhand eines WfMS. Diese umfasst mehrere Komponenten, die für eine reibungslose Nutzung des Systems erforderlich sind. Das 3D-Simulationsprogramm Factory I/O bildet die zentrale Komponente zur Erstellung des digitalen Zwillings der virtuellen Fertigungslinie. Zur Automatisierung geregelter Abläufe

9. Technische Implementierung

in jener wird OpenPLC¹ als PLC-Komponente eingesetzt. Das WfMS CPEE übernimmt die Kontrolle über den digitalen Zwilling und instanziiert prozessbasierte Simulationen, welche über die TWAIN, einer intuitiven Schnittstelle für Nutzer und dem digitalen Zwilling, definiert, verwaltet und ausgeführt werden können. Sämtliche Ereignisse, wie Werteänderungen von Sensoren und Zustände von Maschinen und Geräten, in der virtuellen Fertigungslinie werden in der Echtzeitdatenbank InfluxDB gespeichert und von der Dashboard Komponente Grafana visuell aufbereitet. Die wichtigste generische Komponente in dieser Architektur ist die Simulations-API. Diese wird sowohl vom WfMS als auch von der TWAIN genutzt, um Tasks einzuleiten und abzuschließen, indem benötigte Maschinen und Geräte eingeschaltet werden.

Zur Realisierung des Prototyps wird eine Kombination aus etablierten Drittanbieter-Softwarekomponenten und Eigenentwicklungen verwendet, um den Anforderungen der Implementierung gerecht zu werden. Die folgende Übersicht bietet eine Vorschau auf die in der Implementierung verwendeten Technologien und Programme. Dabei wird jeder Eintrag beschrieben und verweist auf das entsprechende Unterkapitel, welches diesbezüglich weitere Informationen enthält.

1. **OpenPLC** (siehe Kapitel 9.2.1)
 - Automatisierung der virtuellen Fertigungslinie.
2. **Factory I/O** (siehe Kapitel 9.2.2)
 - 3D-Simulationsprogramm zur Abbildung der virtuellen Fertigungslinie
3. **Factory I/O API** (siehe Kapitel 9.2.2)
 - Schnittstelle zum 3D-Simulationsprogramm
4. **Factory I/O SDK** (Eigenentwicklung², siehe Kapitel 9.2.2)
 - Schnittstelle zum 3D-Simulationsprogramm
5. **CPEE** (siehe Kapitel 9.2.3)
 - WfMS zur Steuerung des digitalen Zwillings
6. **InfluxDB** (siehe Kapitel 9.2.4)
 - Echtzeitdatenbank für Ereignisse
7. **Grafana** (siehe Kapitel 9.2.4)
 - Dashboard-Tool

¹<https://openplcproject.com/>

²Eigenentwicklung basiert auf SDK von Real Games

8. **MQTT** (siehe Kapitel 9.2.6)
 - Protokoll zur IoT-Datenübertragung
9. **MQTT Bridge** (Eigenimplementierung, siehe Kapitel 9.2.6)
 - Übermittelt MQTT Topics an den Kafka Broker
10. **Kafka** (siehe Kapitel 9.2.6)
 - Verarbeitung von Datenströmen
11. **Kafka Consumer** (Eigenentwicklung, siehe Kapitel 9.2.6)
 - Speichert Ereignisse in Echtzeitdatenbank
12. **TWAIN** (Eigenentwicklung, siehe Kapitel 9.1.3)
 - Schnittstelle zum digitalen Zwilling
13. **Simulations-API** (Eigenentwicklung, siehe Kapitel 9.1.2)
 - Stellt Services zur Durchführung von Simulationen sowie zur Überwachung der Funktionsfähigkeit der Implementierung und von Werten der Fertigungslinie bereit.
14. **Swagger** (siehe Kapitel 9.2.5)
 - API-Beschreibung

Anmerkung zur Sprachauswahl in der Implementierung

Obwohl diese Arbeit in deutscher Sprache verfasst wurde, erfolgt die technische Umsetzung, insbesondere der Code, in englischer Sprache. Diese Entscheidung basiert auf der gängigen Praxis in der Programmierwelt. Die Verwendung von Englisch als Programmiersprache ermöglicht eine bessere Konsistenz und Kompatibilität mit anderen Entwicklern und Programmen. Darüber hinaus verwendet Factory I/O standardmäßig englische Bezeichnungen für Objekte. Die Verwendung dieser Bezeichnungen erleichtert die Interaktion und das Verständnis innerhalb der Factory I/O-Umgebung. Das Prozessmodell im WfMS und die Web-Anwendung des digitalen Zwillings werden, wie der Rest der Arbeit, für die Nutzer in deutscher Sprache erstellt.

9.1. Umsetzung des Prototyps

In diesem Unterkapitel wird die Umsetzung der abstrakten Architektur in eine konkrete Softwarearchitektur beschrieben. Hierzu wird ein bewährtes Architekturmuster angewendet. Des Weiteren werden Eigenentwicklungen präsentiert und detailliert erläutert, die im Rahmen der Implementierung erstellt wurden. Eine zentrale Rolle spielt hierbei die TWAIN, die sowohl zur Steuerung als auch zur Überwachung des digitalen Zwillings dient. Schließlich wird erläutert, wie die Automatisierung der virtuellen Fertigungslinie

9. Technische Implementierung

umgesetzt sowie Echtzeit-Ereignisse durch eine IoT-End-to-End-Integration übermittelt werden.

9.1.1. Architektur der Implementierung

In diesem Unterkapitel wird der Fokus insbesondere auf die Art der Architektur sowie die im Rahmen dieser verwendeten Technologien und Programme gelegt. Dadurch soll verdeutlicht werden, warum sich jene Architektur und Technologien gerade für diese Implementierung besonders gut eignen. Detailliertere Informationen zu den verwendeten Drittanbieter-Softwarekomponenten befinden sich im Unterkapitel 9.2.

Zusammenfassend umfasst die implementierte Architektur mit Factory I/O³ ein 3D-Simulationsprogramm, welches über eine Web API und ein SDK verfügt. Für die Nutzung letzterer sind sowohl die .NET Runtime als auch das .NET SDK erforderlich. Die Architektur nutzt zudem verschiedene *Docker* Container, von denen einige auf frei verfügbaren Images basieren. Hierbei können in den verschiedenen Containern unterschiedliche Sprachen verwendet werden, was unter anderem ein Vorteil der angewendeten *Microservices-Architektur* darstellt. Für die eigens erstellten Images wurden je nach Anforderung geeignete oder für jenen Anwendungsfall typische Sprachen ausgewählt. Das Zusammenspiel der einzelnen Elemente dieser Architektur lässt sich folgendermaßen erklären: Factory I/O wird genutzt, um eine 3D-Simulation einer Fertigungslinie aufzubauen, in der eine Produktion simuliert werden kann. Die Workflow Engine CPEE⁴ steuert diese Simulation und wird über einen Docker-Container ausgeführt, der über die Web-API und das SDK von Factory I/O mit dieser interagiert. Beide Komponenten (API und SDK) ermöglichen das Ein- und Ausschalten von Geräten und Maschinen sowie das Ändern von Werten solcher. Mit Hilfe des SDK von Factory I/O können Ereignisse wie Änderungen von Sensorwerten entsprechende Funktionen auslösen. Des Weiteren werden Echtzeitdaten gesammelt, welche mit Hilfe von MQTT und Kafka veröffentlicht werden. Ein Dashboard-Microservice speichert die Daten in der Echtzeitdatenbank InfluxDB, die ebenso über einen eigenen Docker-Container läuft. Dabei sind ein Kafka-Broker, ein MQTT-Broker, eine MQTT-Bridge und ein Kafka-Consumer involviert. Abschließend wird ein Dashboard des digitalen Zwillings in einer Webanwendung eingebettet, das aktuelle Daten anzeigt. Zudem ermöglicht jene Anwendung neue Simulationen zu definieren und auszuführen.

Microservice-Architektur

Für die Implementierung wurde eine Microservice-Architektur angewendet, welche zahlreiche Vorteile mit sich bringt. Der Einsatz von Microservice-basierenden Docker-Containern erlaubt es, eine flexible und skalierbare Infrastruktur mit einer hohen Verfügbarkeit und Zuverlässigkeit aufzubauen, welche schnell auf Veränderungen der Anforderungen reagieren kann. Ein weiterer Vorteil der Verwendung von Containern ist die Möglichkeit, IoT-Geräte und -Anwendungen zu imitieren und so eine virtuelle IoT-Umgebung zu schaffen, welche dem Konzept des digitalen Zwillings ähnelt [98]. Um solche virtuelle IoT-Umgebung zu

³<https://factoryio.com/>

⁴<https://cpee.org/>

erstellen, wurden verschiedene Container angelegt, darunter Container für die PLC-Logik, den MQTT-Broker, das Event-Streaming, die Echtzeitbank und das Dashboard. Diese Container sind in der Lage, miteinander zu kommunizieren und Daten auszutauschen, wodurch ein reibungsloses Funktionieren der virtuellen IoT-Umgebung ermöglicht wird. Microservices sind ein modernes Software-Architekturkonzept, welches eine schnelle Entwicklung von neuen Anwendungskomponenten ermöglicht. Im Gegensatz zur traditionellen Monolithic Architektur sind Microservices verteilt und lose gekoppelt, wodurch Änderungen an einem Service vorgenommen werden können, ohne dass die gesamte Anwendung nicht mehr funktioniert. Jeder Microservice kann unabhängig von anderen Microservices entwickelt und bereitgestellt werden sowie alleine und ohne negative Auswirkungen auf andere Services ausgeführt werden. Dies macht Microservices sehr flexibel und skalierbar [99]. Docker ist ein Tool für Entwickler, welches jene dabei unterstützen soll moderne Anwendungen zu erstellen, zu teilen und auszuführen. Ein Docker-Container ist eine leichtgewichtige, eigenständige sowie standardisierte Einheit von Software, die Code und alle damit verbundenen Abhängigkeiten enthält und solchen ausführen kann. Container verfügen über Code, Laufzeitumgebungen, Systemtools, Systembibliotheken und Einstellungen. Weiters können Docker-Container problemlos skaliert werden, um eine größere Anzahl von Anfragen zu verarbeiten. Dies führt dazu, dass Anwendungen schnell und zuverlässig in verschiedenen Umgebungen ausgeführt werden können. Docker eignet sich insbesondere zur Realisierung einer Microservice-Architektur, da jeder Microservice als einzelner Docker-Container bereitgestellt werden kann [100].

Abbildung 9.2 zeigt die detaillierte Architektur der Implementierung. Diese besteht im Wesentlichen aus zwei Teilen:

- **Client Anwendungen und Dienste:** Umfasst mit der TWAIN, Factory I/O und dessen Web API sowie SDK, Anwendungen und Dienste, welche zusätzlich zur Ausführung der Implementierung notwendig sind. Dies lässt sich damit begründen, da Factory I/O nur auf Windows funktioniert und daher nicht in einem Docker Container ausgeführt werden kann. Folglich werden auch die gestartete Web API von Factory I/O sowie das in C# entwickelte SDK auf der Client-Seite ausgeführt. Schließlich muss auch die Web-Anwendung des digitalen Zwillings inkl. Simulation und Dashboard mittels Internet Browser geöffnet werden, um die Funktionalitäten jener nutzen zu können.
- **Docker Host:** Setzt sich aus den im Docker-Compose definierten Containern zusammen. Hierbei werden verfügbare *Images* wie InfluxDB, Grafana, Eclipse Mosquitto, Confluent Zookeeper und Kafka verwendet. Zusätzlich werden mit Hilfe von *Dockerfiles* die WfMS-, Simulationsausführungs-, Frontend-, MQTT Bridge- und PLC-Microservices erstellt.

9. Technische Implementierung

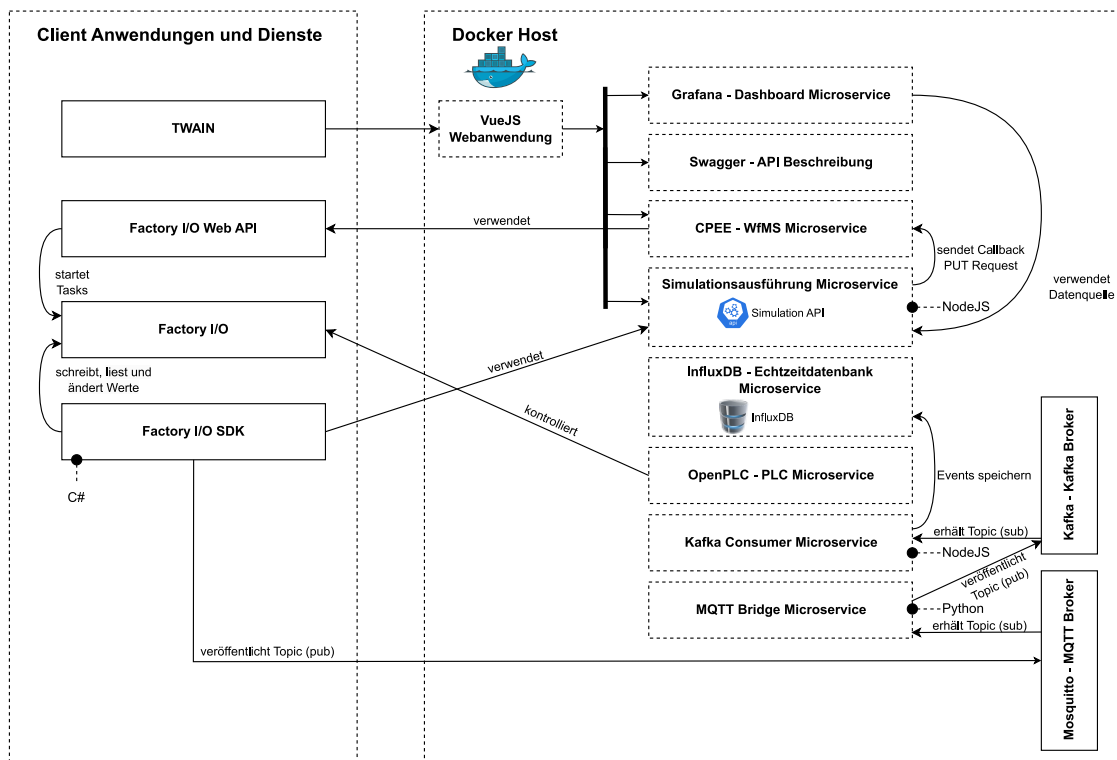


Abbildung 9.2.: Detaillierte Darstellung der Softwarearchitektur

Nachdem alle Docker Container sowie Factory I/O, dessen Web API und SDK gestartet wurden, können anhand der TWAIN Ist-Daten sowie -zustände des digitalen Zwillings eingesehen sowie Simulationen definiert und folglich vom WfMS gestartet werden. Hierzu werden die gesamte 3D-Simulation in Factory I/O vom PLC-Microservice kontrolliert sowie Endpoints der Web API von Factory I/O verwendet, um beliebige Tags abzuändern. Auf diese Weise werden die benötigten Maschinen oder Geräte für den jeweiligen Task zunächst gestartet. Währenddessen bleibt der Task solange aktiv, bis ein entsprechender Sensor ausgelöst wird, indem mit Hilfe des SDK jene über den gesamten Zeitraum hinweg überwacht werden. Sobald ein solches Signal festgestellt wurde, werden die Werte involvierter Sensoren, Geräte und / oder Maschinen als MQTT *Topic* weiter an den MQTT Broker gesendet. Zudem wird ein zugehöriger Endpoint des Simulationsausführung-Microservice verwendet, um folglich einen HTTP PUT Request mit Callback-Parameter an das WfMS zu senden, damit der jeweilige Task abgeschlossen wird. Der MQTT Bridge-Microservice leitet eingehende MQTT-Topics umgehend an den Kafka Broker weiter, wessen Topics wiederum vom Kafka Consumer Microservice abonniert werden. Letzterer speichert schließlich die Werte einzelner Sensoren, Maschinen und Geräte in der Echtzeitdatenbank InfluxDB, welche vom Grafana Dashboard Microservice als Datenquellen verwendet wird.

9.1.2. Simulations-API

Die eigenentwickelte REST-API, entwickelt mit NodeJS und dem ExpressJS-Framework, ermöglicht das Ein- und Ausschalten von Maschinen sowie das Vorbereiten von Materialien auf dem Fließband der virtuellen Fertigungslinie. Darüber hinaus greift die API auf die Factory I/O-API zu, um Echtzeitwerte und Zustände von verschiedenen Sensoren, Geräten und Maschinen abzurufen. Diese Werte und Zustände werden dann den entsprechenden Tasks zugeordnet.

9.1.3. Digital Twin Simulation Application (TWIN)

Bei der Web-Anwendung handelt es sich um eine Eigenentwicklung einer Single-Page-Anwendung, die mit dem JavaScript Framework Vue.js erstellt wurde. Diese verwendet die Simulations-API um Informationen anzuzeigen, Simulationen zu definieren, zu verwalten und zu starten. Weiters bietet sie eine intuitive Benutzeroberfläche mit einem interaktiven Dashboard, welches Echtzeitwerte anzeigt. Gleichzeitig dient die Anwendung zur Dokumentation der gesamten Implementierung.

Zu den Funktionalitäten von TWIN zählen das Erstellen und Verwalten von Simulationen sowie die Instanzieren von Prozessen, wodurch folglich Simulationen durchgeführt werden können. Des Weiteren bietet die Web-Anwendung die Möglichkeit, Echtzeitwerte und Statusinformationen der virtuellen Fertigungslinie zu überwachen und die Überprüfung der ordnungsgemäßen Funktionalität sämtlicher Services der Implementierung sicherzustellen. Schließlich werden die Echtzeitwerte in Form eines Dashboards präsentiert, um den Benutzern eine übersichtliche Darstellung der aktuellen Daten zu bieten. Weitere Details zu den Funktionalitäten von TWIN befinden sich im Unterkapitel 9.3.

9.1.4. Steuerung des digitalen Zwillings durch das WfMS

Zur Integrierung der beiden Technologien werden das verfügbare SDK sowie die Web API von Factory I/O genutzt, wodurch die Kontrolle des digitalen Zwillings anhand eines WfMS realisiert werden kann. Mit der SDK können Entwickler auf die Funktionen und Tools von Factory I/O zugreifen und diese in ihre eigenen Anwendungen integrieren. Die API stellt eine Schnittstelle bereit, über die das WfMS Anfragen an den digitalen Zwilling senden und auf Daten zugreifen kann. Zur Initiierung von Simulationen mit unterschiedlichen Parametern wird eine RESTful-API entwickelt, welche Dienste für die jeweilige Tasks der Produktion bereitstellt und Anfragen an die Web API von Factory I/O sendet. Während der Simulation sendet das WfMS HTTP-Requests an jene API, um Maschinen und andere Tags in Factory I/O einzuschalten sowie produktionsrelevante Informationen zu hinterlegen und solche den entsprechenden Paletten oder Paketen mit Hilfe eines RFID-Lesegerätes zuzuordnen. Weiters werden sämtliche Entscheidungen vom WfMS getroffen, beispielsweise welche Produktionswege genommen oder welche Durchführungsoptionen angewendet werden sollen, indem via Parameter die hierzu notwendigen Informationen an die API weitergegeben werden. Sobald bei der Produktion bestimmte Sensoren ein neues Signal geben, kann in der SDK definiert werden, wie darauf reagiert

9. Technische Implementierung

werden soll. Hierbei werden unter anderem Callback PUT-Requests an die API des WfMS gesendet, um den Prozess fortzusetzen. Dadurch wird in der Workflow-Engine ersichtlich, wo sich der Prozess gerade befindet, indem bei Abschluss eines Tasks automatisch weitergeschaltet wird. Außerdem werden zuvor eingeschaltete Maschinen oder andere Tags wieder ausgeschaltet sowie RFID-Transponder-Werte entweder gelesen oder neu geschrieben.

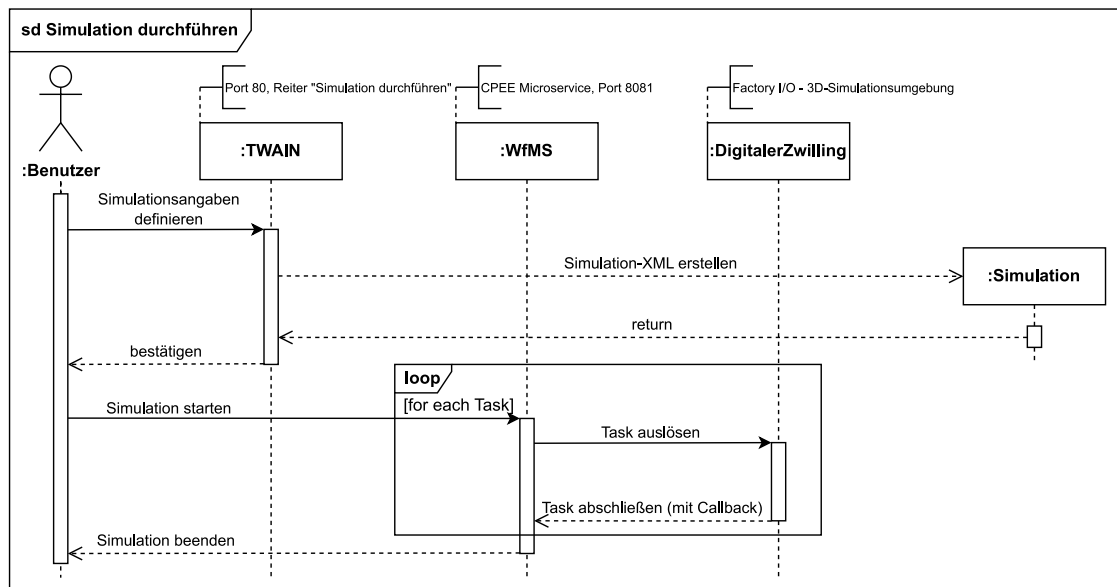


Abbildung 9.3.: Sequenzdiagramm: Simulation erstellen und durchführen

Das Sequenzdiagramm in Abbildung 9.3 zeigt den Ablauf der Erstellung und Ausführung einer Simulation. Der Fokus liegt hierbei auf der Interaktion zwischen dem Benutzer, dem WfMS, dem digitalen Zwilling und der Web-Anwendung. Der Benutzer definiert zunächst über die TWAIN alle notwendigen Angaben für die Simulation. Anschließend erstellt jene ein Extensible Markup Language (XML)-File mit den erforderlichen Informationen, welches dem Benutzer zurückgegeben wird. Optional kann der Benutzer das File herunterladen und speichern. Mittels des WfMS (CPEE) wird die Simulation gestartet. Dabei wird, solange Tasks vorhanden sind, eine Schleife ausgeführt, welche den entsprechenden Task startet, indem die notwendigen Maschinen und Geräte eingeschaltet werden. Sobald ein Callback-Request erfolgt, wird die Maschine wieder ausgeschaltet und es folgt der nächste Task. Nachdem die Schleife vollständig durchlaufen wurde, wird die Simulation beendet.

9.1.5. Automatisierung der Fertigungslinie

Für die Automatisierung einer fiktiven Fertigungslinie wird das Simulationsprogramm Factory I/O eingesetzt. Dabei unterstützt Factory I/O zahlreiche Treiber wie beispielsweise

Modbus TCP/IP Server, der für die Kommunikation mit einem externen Controller verantwortlich ist. OpenPLC erhält Input- und Output-Signale von verschiedenen Geräten wie Sensoren oder Fließbändern und regelt anhand dieser den Ablauf in Factory I/O. Hierfür wird die notwendige Logik mit Hilfe des OpenPLC Editors erstellt und schließlich als strukturierter Text kompiliert. Durch die Verwendung von Factory I/O und OpenPLC können Prozesse in der Fertigungslinie automatisiert und folglich optimiert werden, was zu einer höheren Effizienz und Produktivität führen kann. Die flexible und anpassungsfähige Struktur ermöglicht zudem eine schnelle und unkomplizierte Anpassung an veränderte Anforderungen in der Produktion. Mit dieser Lösung wird eine leistungsfähige und zuverlässige Automatisierung der Fertigungslinie realisiert.

9.1.6. End-to-End IoT Integration

Das MQTT-Protokoll ist ein standardbasiertes Messaging-Protokoll, welches sich bereits zu einem Standard für die IoT-Datenübertragung entwickelt hat. MQTT dient unter anderem zur Kommunikation zwischen Maschinen und liefert insbesondere hinsichtlich Skalierbarkeit, Zuverlässigkeit, Sicherheit sowie Flexibilität beachtliche Vorteile [101]. Laut Bances et al. [102] sei MQTT ein leichtgewichtiges IoT-Protokoll, welches speziell für den Einsatz in Szenarien mit geringer Bandbreite entwickelt wurde. Weiters verwendet MQTT das sogenannte Publish/Subscribe-Kommunikationsmuster (Pub/Sub) und gewährleistet eine effiziente Kommunikation bei geringstmöglichem Batterieverbrauch der Hardwaregeräte. Jenes Muster ermöglicht die massenhafte Verteilung von Informationen von einem Herausgeber an mehrere interessierte Abonnenten, wodurch der Netzverkehr reduziert wird. Hierbei ist es nicht erforderlich, dass Herausgeber einzelne Nachrichten versenden. Stattdessen verwendet man einen Vermittlungsknoten (Broker), um die Informationen zu verteilen. Aufgrund jener Eigenschaften wird MQTT mit Pub/Sub im Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) als vorgeschlagenes Kommunikationsprotokoll aufgeführt, welches die Grundlage zur standardisierten Kommunikation zwischen Verwaltungsschalen bildet [102, 103]. MQTT ist ein Protokoll, das speziell für den Einsatz in unzuverlässigen Netzwerken und einer hohen Anzahl von Geräten entwickelt wurde. Im Gegensatz dazu ist Apache Kafka eine Open-Source-Plattform für verteiltes Event-Streaming, die von Tausenden von Unternehmen zur Erstellung von leistungsstarken Datenpipelines, Streaming-Analytik, Datenintegration und geschäftskritischen Anwendungen genutzt wird. Mehr als 80% der Fortune-100-Unternehmen setzen auf Kafka. Die Plattform verfügt über mehrere Kernkompetenzen wie Skalierbarkeit, hohe Durchsatzrate, permanente Speicherung, hohe Verfügbarkeit und eine integrierte Stream-Verarbeitung [104, 105]. In einigen IoT-Projekten wird MQTT mit Kafka kombiniert, da die Vor- und Nachteile beider Technologien sehr gut ergänzt werden. Die Vorteile von Kafka sind insbesondere die zahlreichen Stream-Verarbeitungsmöglichkeiten, Skalierbarkeit, Verfügbarkeit, die langfristige Speicherung und Wiederaufbereitung von Ereignissen sowie Integration mit anderen Unternehmens-Technologien. MQTT hingegen ist leichtgewichtig, unterstützt alle Programmiersprachen, wurde für schwache Verbindungen und Hochlatenzszenarien wie mobile Netzwerke entwickelt, ist ein ISO-Standard und das beliebteste IoT-Protokoll. Allerdings bietet MQTT nur die Pub/Sub-Funktionalität und keine Stream-Verarbeitung

9. Technische Implementierung

oder Wiederaufbereitung von Ereignissen. Kafka hingegen ist nicht für Tausende von Verbindungen ausgelegt und erfordert eine stabile Netzwerk- und Infrastruktur. Gemeinsam ermöglichen MQTT und Kafka eine Echtzeit-Integration, Verarbeitung und Analyse von IoT-Daten. Ein weiterer Standard für IIoT ist Open Platform Communications Unified Architecture (OPC-UA), der von den meisten neuen Maschinen, Geräten und Middleware in IIoT-Umgebungen unterstützt wird [104]. Gemeinsam bilden MQTT und Apache Kafka eine perfekte Kombination für die End-to-End-Integration von IoT-Geräten bis hin zum Rechenzentrum. Für jene Kombination gibt es bereits einige Implementierungsansätze [106]. Für den angewendeten Ansatz⁵ in der technischen Implementation werden sämtliche Daten von Sensoren und sonstigen Geräten mit Hilfe eines MQTT-Producers an einen MQTT-Broker weitergegeben. Anschließend sorgt eine MQTT Bridge für eine erfolgreiche Integration beider Technologien. Jene Bridge erfordert sowohl einen MQTT-Client, um sich mit dem MQTT-Broker zu verbinden, sowie einen Kafka-Client, um sich mit dem Kafka-Broker zu verbinden. Schließlich können relevante Topics abonniert und empfangen werden.

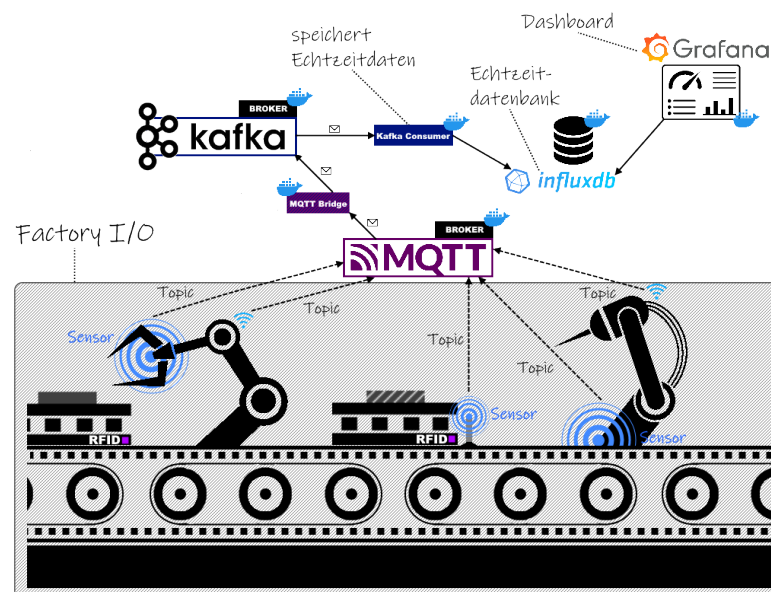


Abbildung 9.4.: End-to-End IoT Integration in der Implementierung

Abbildung 9.4 zeigt die implementierte End-to-End Integration zur Kommunikation zwischen den einzelnen Maschinen, Geräten sowie Sensoren und dem digitalen Zwilling. Während der Simulationsabwicklung werden mit Hilfe der SDK von Factory I/O sowohl RFID-Transponder-Werte gelesen sowie Änderungen von IoT-Geräten- bzw. Maschinenzuständen oder Sensorenwerten als MQTT Topics an den MQTT Broker weitergegeben. Jene werden anschließend über die MQTT Bridge weiter an den Kafka Broker übertragen.

⁵<https://medium.com/python-point/mqtt-and-kafka-8e470eff606b>

Schließlich abonniert ein Kafka Consumer jene Topics und erhält sowie speichert jene in der Echtzeitdatenbank InfluxDB. Zur grafischen Darstellung in Form eines Dashboards wird Grafana genutzt, welches jene Datenbank als Datenquelle verwendet.

9.2. Eingesetzte Programme und Technologien

In diesem Unterkapitel werden die verwendeten Drittanbieter-Softwarekomponenten vorgestellt und die Konfigurationen solcher beschrieben.

9.2.1. OpenPLC

Bei OpenPLC handelt es sich, sowohl in Software als auch in Hardware, um eine vollständig funktionsfähige und standardisierte Open-Source Lösung einer SPS. Laut der offiziellen Projektwebseite von OpenPLC gehören unter anderem die Industrie- und Home-Automatisierung sowie das IoT zu deren Haupteinsatzgebieten. Mit Hilfe der beiden OpenPLC-Bestandteile Runtime und Editor können Nutzer PLC Programme entwickeln und diese schließlich ausführen. Außerdem unterstützt der OpenPLC Editor fünf verschiedene Programmiersprachen speicherprogrammierbarer Steuerungen: Kontaktplan (KOP, engl.: Ladder Diagram, LD), FUP (engl.: Function Block Diagram, FBD), Anweisungsliste (AWL, engl.: Instruction List, IL), ST (engl.: Structured Text) und Ablaufsprache (AS, engl.: Sequence Function Chart, SFC) [107]. Man unterscheidet diese Programmiersprachen in zwei verschiedene Kategorien: textuelle Sprachen (AWL, ST) sowie grafische Sprachen (AS, KOP, FUP) [108]. Aus der Nutzung des OpenPLC Editors geht hervor, dass gleich nach dem Anlegen eines neuen Projektes die Möglichkeit besteht, Programme sowie Funktionen in einer sowie mehreren unterschiedlichen Sprachen zu erstellen. So können unter anderem ST-Funktionen in FUP-Programmen ohne Schwierigkeiten genutzt werden. Neben dem OpenPLC Editor, welcher das Schreiben, Kompilieren sowie die Bereitstellung in der OpenPLC Runtime ermöglicht [107]^a, werden mit Hilfe der OpenPLC Runtime PLC Programme ausgeführt. Hierbei besteht die Möglichkeit einige Parameter der Runtime über einen eingebauten Webserver zu konfigurieren [107]^b. Folglich soll im Rahmen der Masterarbeit die OpenPLC Runtime durch ein Slave Gerät erweitert werden. Hierbei bestehe laut OpenPLC unter anderem die Möglichkeit diverse Modbus Slave Geräte einzusetzen [107]^c. Im Rahmen der Implementierung gehört OpenPLC zu den Drittanbieter-Softwarekomponenten, welche lediglich beim Start konfiguriert werden. Hierzu werden das im OpenPLC Editor erstellte ST-Programm geladen sowie ein Slave Gerät erstellt.

Abbildung 9.5 zeigt einen Ausschnitt des FUPs, mit dem OpenPLC den ST-Code generiert. Auf der rechten Seite wird mit Hilfe des Funktionsblocks *CTUD* ein Zähler erstellt, der

Betrifft folgende Unterseite von [107]:

^aOpenPLC Editor Overview

^bOpenPLC Runtime Overview

^cSlave Devices

9. Technische Implementierung

den Wert von *start_counter* erhöht, wenn der Startsensor ausgelöst wird. Dies geschieht, wenn eine neue Palette auf dem Fließband (*Conveyor*) platziert wird. Sobald die Palette einen bestimmten Punkt erreicht, löst einer von zwei möglichen Sensoren eine Subtraktion um eins aus. Auf der linken Seite der Abbildung wird definiert, dass das Fließband immer dann eingeschaltet ist, solange der Zähler größer als null ist.

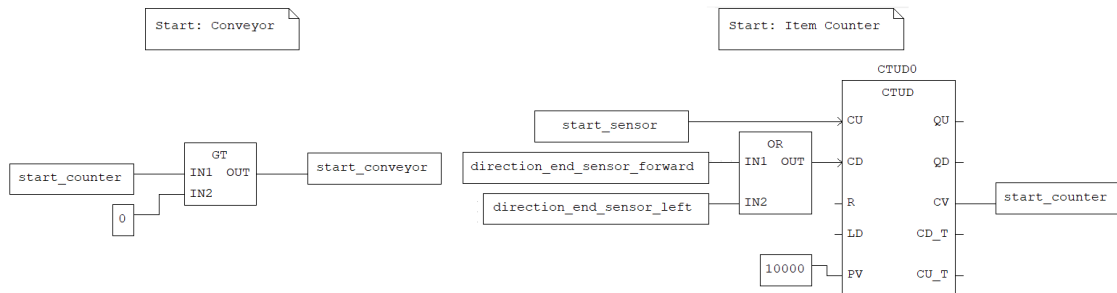


Abbildung 9.5.: FUP Ausschnitt in OpenPLC (Screenshot)

9.2.2. Factory I/O

Im Rahmen der folgenden Masterarbeit soll die Drittanbieter-Softwarekomponente Factory I/O in Kombination mit dem I/O Treiber Modbus TCP/IP Server eingesetzt werden, welcher einen sogenannten „Slave“ implementiert [97]^d. Modbus Slave Geräte werden von OpenPLC unterstützt und können zur OpenPLC Runtime angefügt werden. Somit wird die Anzahl von I/O Punkten erweitert [107]^f.

In Anbetracht der Thematik digitaler Zwillinge bietet Factory I/O insbesondere durch einer Variation an vorgefertigter und beliebig einsetzbarer Sensoren die Möglichkeit, die Präsenz, die Distanz sowie den Typ von Dingen zu ermitteln [97]^e. Sämtliche Sensoren sowie Antriebe in Factory I/O verfügen über einen oder mehrere Tags, welche die Werte von Antrieben und Sensoren mit einem Controller verknüpfen sowie die Möglichkeiten bieten, Antriebe manuell zu steuern. Tags setzen sich hierbei aus einem Namen sowie einem Wert zusammen. Factory I/O unterstützt insgesamt drei verschiedene Datentypen bei Tag-Werten: Boolean, Float sowie Integer [97]^f. Zur Abfrage sowie zum Setzen von Tag-Werten bietet Factory I/O sowohl ein SDK als auch eine Web API, welche über einen umfangreichen Bestand an Services verfügt [97]^g.

Betrifft folgende Unterseite von [97]:

^dModbus-Server

^eParts/Sensors

^fTags

^gWeb-API

Factory I/O API und SDK

Bei der Implementierung werden zwei bereitgestellte Schnittstellen von Factory I/O genutzt, um mit der Simulationsumgebung zu interagieren. Dabei werden die verfügbaren Services der API verwendet. Mithilfe des SDKs kann eine individuelle Anwendung in C# entwickelt werden, um auf die Ein- und Ausgabepunkte der Simulation zuzugreifen.

9.2.3. Cloud Process Execution Engine als WfMS

Im Rahmen der Abschlussarbeit soll mit Hilfe von CPEE, einer kostenlosen sowie modularen und serviceorientierten Ausführungseinheit, ein zugrunde liegender BPMN-Prozess in jener abgebildet und ausgeführt werden. Schließlich sollen Instanzen von jenem Prozess, welche jeweils eine Bestellung darstellen, mit unterschiedlichen Parametern via der Ausführungseinheit ausgeführt werden können. Somit soll sich unter anderem die Abfolge sowie die Art und Weise der Fertigung in Factory I/O unterscheiden. Dies kann unter anderem dadurch realisiert werden, da CPEE das Hinzufügen von REST-Diensten unterstützt [84]. Um Gegenstände bzw. Bestandteile einer Bestellung während des Fertigungsprozesses zuordnen zu können, bietet Factory I/O die Möglichkeit RFID Tags einzusetzen, welche über eine einzigartige Seriennummern verfügen [97]^b. Jene wird schließlich anhand eines Callbacks zurück an das WfMS weitergeleitet und hinterlegt. Auf diese Weise können vom WfMS ausgehende Requests an die Simulations-API denjenigen Objekten zugeordnet werden, die sich zu diesem Zeitpunkt in der Simulation befinden und über übereinstimmende Seriennummern verfügen. Im Rahmen der Implementierung wird die Open-Source-Ausführungseinheit CPEE in einem Docker-Container gestartet und deren API genutzt.

9.2.4. Echtzeitdatenbank und Dashboard

Im Folgenden wird beschrieben, welche Drittanbieter-Softwarekomponenten zur Erstellung des Dashboards verwendet werden. Beim Start des Docker-Containers werden die beiden eingesetzten Komponenten konfiguriert. Eine wichtige Komponente ist die dafür benötigte Datenbank. Hierzu wird in der Implementierung InfluxDB als eine geeignete Lösung betrachtet. Folglich werden die Daten jener über Grafana in Form eines Dashboards angezeigt, welches schließlich in die Web-Anwendung eingebettet wird.

InfluxDB

InfluxDB⁶ ist eine Smart Data Plattform, die speziell für die Verwaltung von Zeitreihendaten entwickelt wurde. So ermöglicht diese unter anderem ein effizientes Management von Zeitreihendaten und Durchführung von Datenanalysen in Echtzeit sowie eine high-speed Erfassung riesiger Datenmengen. Als Open-Source-Produkt bietet InfluxDB eine community-unterstützte Zeitreihendatenbank, die auf eigenen Endgeräten oder Servern betrieben werden kann. Dies ermöglicht Benutzern die volle Kontrolle über ihre Daten

⁶<https://www.influxdata.com/>

9. Technische Implementierung

und die Flexibilität, InfluxDB in ihrer eigenen Infrastruktur zu betreiben [109]. Zudem verfügt InfluxDB über eine Vielzahl von APIs und Tools, die den Einstieg und die Verwendung erleichtern [110]. InfluxDB ist eine bewährte Lösung für die Verwaltung von Zeitreihendaten, insbesondere im Kontext des IoT. IoT-Geräte generieren große Mengen von Zeitreihendaten, und angesichts des wachsenden Bedarfs an fortschrittlicher Datenanalytik benötigt man hierfür eine leistungsfähige Plattform. InfluxDB hat sich als eine solche Plattform etabliert und zählt weltweit anspruchsvolle Kunden wie Siemens und PTC zu seinen Nutzern. Eine der Stärken von InfluxDB liegt in dessen Skalierbarkeit, die es ermöglicht, große Datenmengen aus IoT-Geräten effizient zu verwalten. Anhand der flexiblen und leistungsfähigen Abfragesprache Flux bietet InfluxDB eine geeignete Lösung für den Aufbau von IoT-Lösungen, um Daten zu gestalten und abzufragen. Ein weiteres Merkmal von InfluxDB ist seine Fähigkeit, metadatenbasierte Kontextinformationen zu nutzen, um Erkenntnisse über vergangene, aktuelle und zukünftige Leistung von Anlagen und Prozessen abzuleiten. Darüber hinaus ist InfluxDB auch flexibel in Bezug auf die Unterstützung von Datenformaten. Nicht unterstützte Datenformate können in InfluxDB aufgenommen werden, was es ermöglicht, eine Vielzahl von Datenquellen in IoT-Anwendungen zu integrieren [111].

Grafana als Dashboard

Grafana⁷ ist ein Open-Source-Tool, womit Benutzer sämtliche Daten einer Organisation unabhängig von ihrem Speicherort abfragen sowie visualisieren und Alarmer stellen können. Weiters dient Grafana dem Zweck, Daten zu analysieren und zu verstehen. Dabei ist einer der Hauptvorteile von Grafana die Fähigkeit, Daten aus verschiedenen Quellen zu integrieren. Anstatt auf eine einzige Datenbank angewiesen zu sein, können demnach Daten aus verschiedenen Quellen wie InfluxDB, PostgreSQL und vielen anderen herangezogen werden. Ebenso können mittels Grafana flexible und vielseitige Dashboards erstellt werden sowie mit Hilfe von Transformationen, Datenquellen umbenannt, zusammengefasst, kombiniert und Berechnungen durchgeführt werden. Da Grafana mit dem Prinzip, dass jeder in einer Organisation Zugriff auf Daten haben sollte, entwickelt wurde, haben alle Benutzer die Möglichkeit, eigene Dashboards zu erstellen und schließlich zu teilen [112]. In Grafana können mehrere InfluxDB-Datenquellen definiert werden, wie in der offiziellen Dokumentation von Grafana beschrieben [113]. Im Rahmen der Implementierung wird Grafana ein bereits existierendes Dashboard übergeben, welches schließlich in der TWAIN eingebunden wird.

9.2.5. Swagger UI

Im Rahmen der Implementierung gehört Swagger⁸ zu den Drittanbieter-Softwarekomponenten und dient der Beschreibung der Simulations-API. Swagger ist eine Open-Source-Lösung, die es Entwicklerteams und Endbenutzern ermöglicht, die Ressourcen einer API zu vi-

⁷<https://grafana.com/>

⁸<https://swagger.io/>

sualisieren und damit zu interagieren. Beim Start des Docker-Containers werden hierfür erforderliche OpenAPI-Spezifikationen an Swagger übergeben [114].

9.2.6. Kafka und MQTT

Bei der Implementierung werden zwei Drittanbieter-Softwarekomponenten verwendet. Anhand verfügbarer Docker-Images wird sowohl ein Kafka⁹- als auch ein MQTT¹⁰-Broker erstellt. Um eine nahtlose Integration zu ermöglichen, wird eine eigens entwickelte MQTT-Bridge eingesetzt. Diese sorgt dafür, dass die von IoT-Geräten übermittelten Topics über den MQTT-Broker an den Kafka-Broker weitergeleitet werden. Ein in NodeJS eigenentwickelter Consumer kümmert sich schließlich um eingehende Nachrichten und speichert diese in der Echtzeitdatenbank.

9.3. Aufbau von TWAIN

In diesem Kapitel werden die Unterseiten der digitalen Web-Anwendung des digitalen Zwillings vorgestellt. Generell dient jene als Schnittstelle zwischen Nutzern und dem digitalen Zwilling der virtuellen Fertigungsline. Anhand der Anwendung können Zustände und Werte überwacht, Simulationen definiert, verwaltet und durchgeführt sowie Einblicke anhand eines Dashboards erhalten werden.



Abbildung 9.6.: TWAIN Startseite (Screenshot)

⁹<https://kafka.apache.org/>

¹⁰<https://mosquitto.org/>

9. Technische Implementierung

Die Abbildung 9.6 veranschaulicht die einladende Startseite der Web-Anwendung des digitalen Zwillings. Sie dient als Übersicht, anhand welcher Informationen der Abschlussarbeit entnommen und heruntergeladen werden können. Des Weiteren bietet sie Verweise auf das GitHub-Repository sowie ein Simulations-Aufnahme, welches zeigt, wie Simulationen mit Hilfe der TWAIN durchgeführt werden können. Die Navigationsleiste zeigt die vielfältigen Funktionen der Anwendung, welche im Folgenden präsentiert und ausführlich erläutert werden.

9.3.1. Unterseite: Healthcheck

Diese Unterseite hat die Aufgabe, die ordnungsgemäße Funktionalität aller Services zu überprüfen.

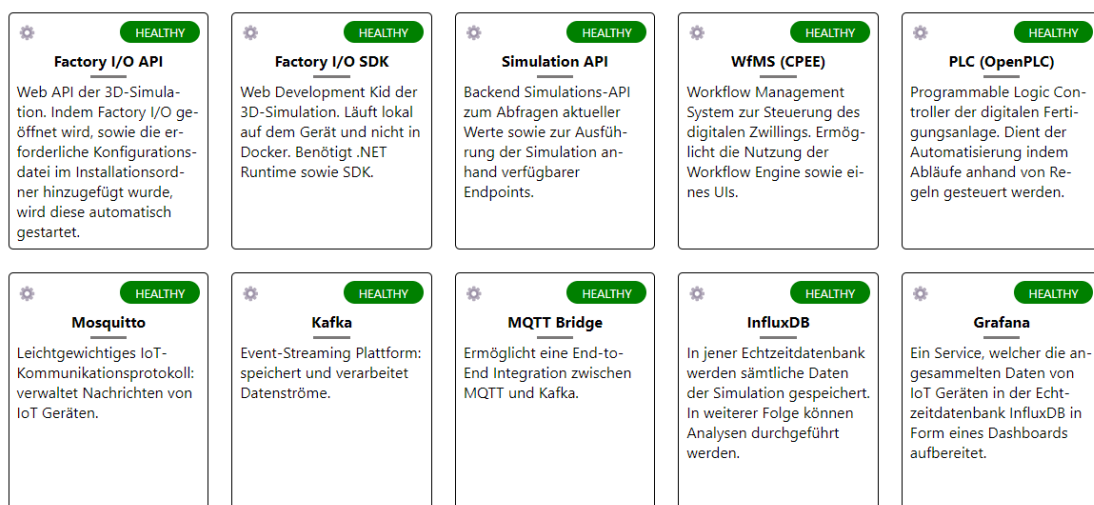


Abbildung 9.7.: TWAIN Healthcheck (Screenshot)

Abbildung 9.7 zeigt die Healthcheck-Unterseite der Webanwendung des digitalen Zwillings. Hier können Informationen über die Services sowie deren Status abgerufen werden, um festzustellen, ob sie wie erwartet funktionieren. Bei Problemen mit einem Service werden entsprechende Tipps zur Behebung angezeigt.

9.3.2. Unterseite: Simulation erstellen

Das Formular auf dieser Unterseite ermöglicht es den Benutzern anhand ihrer benutzerdefinierten Spezifikationen eine Simulation zu definieren. Jene Simulationen basieren auf einen Prozess und treffen eine Reihe von Entscheidungen, die auf den in den Spezifikationen enthaltenen Bedingungen beruhen. Diese Bedingungen werden verwendet, um die erforderlichen Parameter und Einstellungen festzulegen, die für die Durchführung und Ablaufregelung der Simulation erforderlich sind.

Simulation Name (überschreibt bereits existierende Simulationen mit demselben Namen)
Simulation_S1

Was für ein Produkt soll produziert werden?
Schreibtisch

Wähle einen Tisch-Stil aus
Modern

Paket abhängig von Höhe für Versand kategorisieren:
Die Grenze liegt bei ≥ 200 cm
Kleines Paket Großes Paket

Paket abhängig von Gewicht für Versand kategorisieren:
Die Grenze liegt bei ≥ 500 g
Kleines Paket Großes Paket

☐ **Zusatzrüstung** (optional)
☐ **Expresslieferung** (optional)

Paketgrößen festlegen:

Box S	Box M	Box L (else-Bedingung)
☒ Schrank	☒ Schrank	☐ Schrank
☐ Zusatzrüstung (erforderlich)	☒ Zusatzrüstung (erforderlich)	☐ Zusatzrüstung (erforderlich)
☐ Schreibtisch	☒ Schreibtisch	☐ Schreibtisch
☐ Zusatzrüstung (erforderlich)	☐ Zusatzrüstung (erforderlich)	☐ Zusatzrüstung (erforderlich)

Simulation erstellen

Abbildung 9.8.: Formular der TWAIN zur Definition von Simulationen (Screenshot)

Abbildung 9.8 zeigt das Formular der TWAIN, das es den Nutzern ermöglicht, neue Simulationen zu definieren. Dabei können Nutzer unter anderem die Produktart und weitere Produktionsspezifikationen festlegen. Es ist auch möglich, Entscheidungsgrundlagen für den Prozess anzugeben, wie z.B. ab welchem Gewicht ein Paket als großes Paket eingestuft wird oder welche Verpackung für bestimmte Kombinationen verwendet werden soll. Schließlich kann die definierte Simulation im Reiter *Simulation durchführen* instanziiert und durchgeführt werden. In diesem Reiter besteht auch die Möglichkeit, die Simulation herunterzuladen oder zu löschen.

9.3.3. Unterseite: Geräte Übersicht

Diese Unterseite dient zur Überwachung von Echtzeitdaten wie Sensorenwerte und Zustände von Maschinen oder Geräten der virtuellen Fertigungslinie.

9. Technische Implementierung

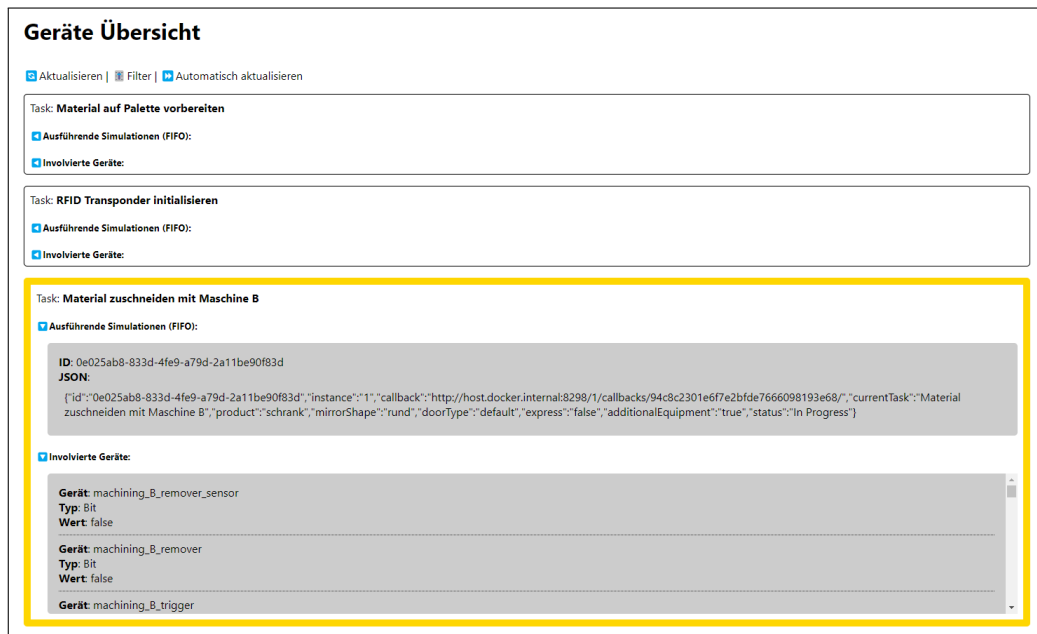


Abbildung 9.9.: TWAIN Geräte Übersicht (Screenshot)

Abbildung 9.9 zeigt die Geräte Übersicht der Webanwendung des digitalen Zwilling. Hier werden Echtzeit-Zustände aller Sensoren, Maschinen und Geräte in Factory I/O dargestellt. Diese Objekte sind nach den jeweiligen Tasks gruppiert. Weiters werden Tasks optisch hervorgehoben, welche gerade ausgeführt werden. Zusätzlich werden Details der Simulation als JSON angezeigt.

9.3.4. Unterseite: Simulation durchführen

Diese Unterseite dient zur Instanziierung und Durchführung prozessbasierter Simulationen anhand des WfMS.

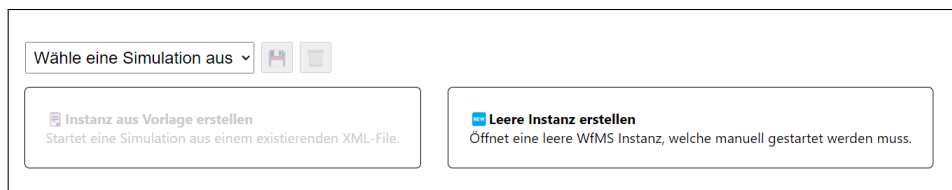


Abbildung 9.10.: Durchführung von WfMS-Instanzen in der TWAIN (Screenshot)

Abbildung 9.10 zeigt die Unterseite, welche es Nutzern erlaubt, neue Instanzen zu erstellen oder existierende Vorlagen zu starten, herunterzuladen oder zu löschen. Sobald eine Instanz gestartet wurde, wird die CPEE-Instanz als *iframe* angezeigt. Außerdem besteht die Option, Instanzen entweder zu schließen oder mehrere gleichzeitig laufen zu lassen.

9.3.5. Unterseite: Dashboard

Diese Unterseite stellt den Nutzern ein Dashboard zur Verfügung, das eine visuelle Darstellung von Echtzeitdaten bietet. Das Dashboard ermöglicht es den Benutzern, die aktuelle Situation der Fertigungslinie schnell zu erfassen. Durch die Darstellung der Echtzeitdaten auf einen Blick können die Nutzer wichtige Informationen über den Status der Anlage, die Leistungskennzahlen oder andere relevante Metriken erfassen, ohne in detaillierte Berichte oder komplexe Datenanalysen eintauchen zu müssen.

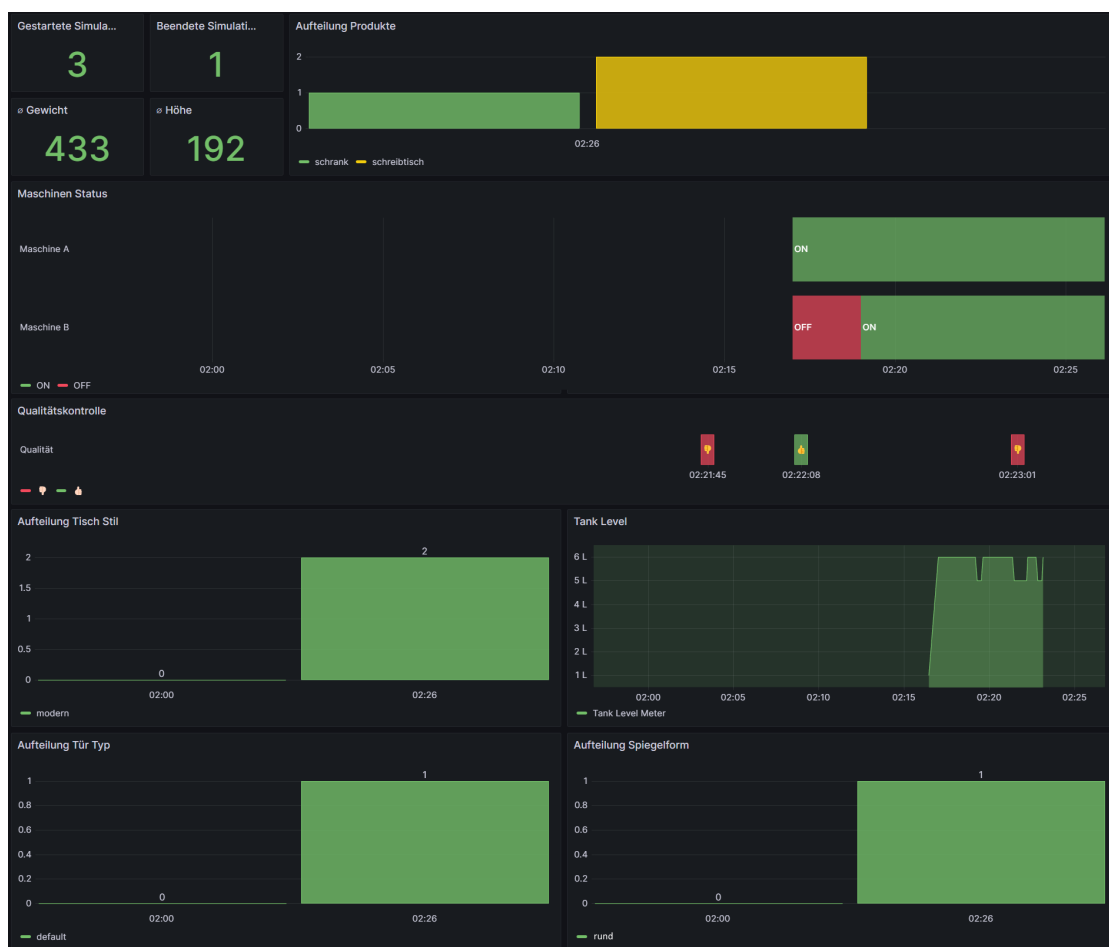


Abbildung 9.11.: Dashboard des digitalen Zwillings in der TWAIN (Screenshot)

Abbildung 9.11 zeigt das Dashboard des digitalen Zwillings, welches mit Hilfe von Grafana erstellt und in der Web-Anwendung eingebettet wird. Dabei nutzt Grafana die Echtzeitdatenbank InfluxDB als Datenquelle, um kontinuierlich Daten abzufragen. Hierbei wird die

9. Technische Implementierung

Abfragesprache Flux¹¹ verwendet. Anhand des Dashboards sollen Nutzer auf einen Blick in der Lage sein, einen schnellen Überblick über die aktuelle Lage der Fertigungslinie zu erhalten. Das Dashboard enthält verschiedene Visualisierungen wie Balkendiagramme, Zeitreihen, Statuszeitleisten und -verläufe sowie Statistiken in Form einzelner Werte. Zudem bietet es eine Übersicht über zahlreiche weitere Informationen. Es zeigt an, wie viele Simulationen gestartet und beendet wurden, die Aufteilung der produzierten Produkte und weitere Aufschlüsselungen nach Tischstil, Spiegelform sowie Türart. Außerdem wird ein Graph des Tanklevels angezeigt, der während der Oberflächenbehandlung abnimmt und anschließend wieder zunimmt. Darüber hinaus ist es möglich, Durchschnittswerte wie Gewicht und Höhe der Pakete einzusehen und den Verlauf des Qualitätskontrollstatus anzuzeigen. Weiters werden die letzten Ergebnisse der Qualitätskontrolle sowie der Status der Maschinen dargestellt.

9.4. Installations- und Konfigurationsanleitung

9.4.1. Systemvoraussetzungen und Tools

Um die Ausführung der Implementierung zu erleichtern, wird Docker¹² verwendet, welches sich um alle erforderlichen Softwarekomponenten, Bibliotheken und Einstellungen kümmert. Da allerdings Factory I/O ausschließlich auf Windows (mind. Windows 7) läuft, wird die im Rahmen dieser Arbeit angewendete Konfiguration für Windows-Geräte in diesem Unterkapitel vorgestellt. Für Benutzer mit anderen Betriebssystemen wird im offiziellen Forum von Factory I/O die Möglichkeit erwähnt, eine virtuelle Maschine (VM) für Factory I/O zu verwenden¹³. Hierbei ist wichtig sicherzustellen, dass alle Treiber auf dem Host aktualisiert sind und die Option für die 3D-Grafikbeschleunigung in den VM-Gast-Einstellungen aktiviert ist sowie ausreichend VRAM (mind. 512 MB) zur Verfügung steht. Zusätzlich zur Docker-Implementierung sind neben Factory I/O auch die aktuellste .NET-Runtime und das .NET-SDK¹⁴ erforderlich, damit das SDK von Factory I/O ausgeführt werden kann. Es ist zudem notwendig, dass sämtliche Komponenten auf demselben System bzw. VM vorhanden sind, um die ordnungsgemäße Ausführung der Implementierung zu gewährleisten. Damit der Code via Terminal heruntergeladen werden kann, muss ebenso das Versionsverwaltungssystem Git¹⁵ installiert werden.

9.4.2. Zugriff und Ausführung des Codes

Der Code der Implementierung ist auf GitHub¹⁶ verfügbar und ermöglicht es Entwicklern, das Projekt einfach zu reproduzieren und bei Bedarf zu erweitern. Um den Code herunterzuladen muss zunächst das GitHub Repository *geklont* werden. Codeausschnitt 9.1 zeigt

¹¹<https://docs.influxdata.com/influxdb/cloud/query-data/get-started/>

¹²<https://www.docker.com/products/docker-desktop/>

¹³<https://community.factoryio.com/t/factory-i-o-installation-at-virtual-machine/657>

¹⁴<https://dotnet.microsoft.com/en-us/download/visual-studio-sdks>

¹⁵<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

¹⁶<https://github.com/lollo1111/Masterarbeit-LP>

dementsprechend, wie ein solcher Clone-Vorgang via Konsole ausgeführt werden kann. Damit das Docker-Compose erfolgreich erstellt werden kann, ist es zwingend notwendig, dass Docker vorher erfolgreich gestartet wird. Anschließend muss lediglich vom aktuellen Verzeichnis in das Projektverzeichnis gewechselt werden, damit schließlich alle Container mit Hilfe von Docker erstellt und gestartet werden können. Davor sind jedoch noch einige Anpassungen erforderlich, um sicherzustellen, dass sowohl die Simulationen als auch der digitale Zwilling korrekt ausgeführt werden können.

```
1 git clone https://github.com/lollo1111/Masterarbeit-LP.git
2 cd Masterarbeit-LP
3 # !Wichtig!
4 # Vor dem ersten Start sowie wenn der PC neu hochgefahren wurde, ist eine
  Anpassung des .env sowie mbconfig.cfg Files notwendig.
5 docker-compose up --build -d
```

Codeausschnitt 9.1: Docker-Compose starten (Terminal)

Erforderliche Anpassungen

Die hier beschriebenen Anpassungen sind sowohl vor dem ersten Ausführen sowie nach einem Neustart des Gerätes **immer** erforderlich. Nachdem der Code erfolgreich heruntergeladen wurde, sowie in das korrekte Verzeichnis gewechselt wurde, ist jenes wie folgt strukturiert:

```
Masterarbeit-LP
├── backend/
├── bridge/
├── cpee/
├── data_stream/
├── factoryio/
├── mosquito/
├── swagger/
├── openplc/
├── ...
├── mbconfig.cfg
├── .env
└── docker-compose.yml
```

Als erstes muss das `.env`-File (Codeausschnitt 9.2) abgeändert werden. Ersetze hierfür den Pfad zur ausführbaren Datei von Factory I/O (`.exe`). Diesen findet man wie folgt:

1. Via *Start* bzw.  nach *Factory I/O* suchen
2. Sobald Factory I/O gefunden wurde, muss der Dateispeicherort via Rechtsklick geöffnet werden.
3. Im Dateispeicherort befinden sich zwei Dateien. Anschließend muss anhand eines Rechtsklicks der Dateipfad geöffnet werden.

9. Technische Implementierung

- Schließlich kann der Pfad durch einen erneuten Rechtsklick auf die Datei *Factory I/O.exe* über *als Pfad kopieren* in der Zwischenablage gespeichert und im vorliegenden File eingefügt werden.

```
1 INFLUXDB_USERNAME=admin
2 INFLUXDB_PASSWORD=adminpassword
3 INFLUXDB_TOKEN=tokenforinfluxdb
4 INFLUXDB_ORG=Uni Wien
5 INFLUXDB_BUCKET=FactoryIO
6 MODBUS_SERVER=172.26.96.1
7 MODBUS_PORT=502
8 FACTORY_IO="C:\PATH\TO\Factory IO.exe"
```

Codeausschnitt 9.2: .env File anpassen

Zusätzlich muss der Wert für *MODBUS_SERVER* aktualisiert werden. Hierzu können in Factory I/O die Treibereinstellungen via **F4** geöffnet werden. Abbildung 9.12 zeigt vergebene Ein- und Ausgaben sowie die Adresse des Modbus Servers. Um die Anpassungen abzuschließen, muss jene Adresse (**ohne** Port) ins File übertragen werden. Hierbei gilt zu beachten, dass diese bei jedem Neustart neu vergeben wird. Die restlichen Angaben können unverändert bleiben.



Abbildung 9.12.: Factory I/O Modbus Server Adresse (Screenshot)

Als nächstes muss das *mbconfig.cfg* File (Codeausschnitt 9.3) vor der ersten Ausführung angelegt bzw. nach einem Neustart aktualisiert werden. Hierbei muss wie beim .env File die aktuelle Modbus Server Adresse in *device0.address* übertragen werden. Das zugrunde liegende .cfg-File erstellt beim Starten des Containers automatisch ein Slave-Device. Jener Ansatz stammt aus einem öffentlich verfügbaren Repository ¹⁷.

```
1 Num_Devices = "1"
2 Polling_Period = "100"
3 Timeout = "1000"
4 device0.name = "FactoryIO"
5 device0.slave_id = "1"
6 device0.protocol = "TCP"
7 device0.address = "172.26.96.1"
8 device0.IP_Port = "502"
9 device0.RTU_Baud_Rate = "None"
```

¹⁷<https://github.com/jpaffrath/docker-openplc>

```

10 device0.RTU_Parity = "None"
11 device0.RTU_Data_Bits = "None"
12 device0.RTU_Stop_Bits = "None"
13 device0.RTU_TX_Pause = "None"
14 device0.Discrete_Inputs_Start = "0"
15 device0.Discrete_Inputs_Size = "128"
16 device0.Coils_Start = "0"
17 device0.Coils_Size = "128"
18 device0.Input_Registers_Start = "0"
19 device0.Input_Registers_Size = "32"
20 device0.Holding_Registers_Read_Start = "0"
21 device0.Holding_Registers_Read_Size = "32"
22 device0.Holding_Registers_Start = "0"
23 device0.Holding_Registers_Size = "32"

```

Codeausschnitt 9.3: mbconfig.cfg File erstellen bzw. anpassen

Damit Factory I/O automatisch das korrekte File öffnet sowie im richtigen Modus gestartet wird, muss ins richtige Verzeichnis navigiert und eine Datei erstellt werden.

1. Verwende  +  und suche nach *C:\ProgramData*. Alternativ kann via dem *Explorer* das Laufwerk C: geöffnet werden. In diesem befindet sich, insofern Factory I/O installiert wurde, das Verzeichnis *ProgramData*, welches wiederum den Ordner *Real Games* beinhaltet.
2. Sobald der Ordner erfolgreich geöffnet wurde, muss in diesem ein File mit dem Namen *auto.cfg* erstellt werden, welches wie folgt aussehen muss:

```

1 license.share_code='licensenumber
2 '
3 ui.show_welcome_window = False
4 scene.load_from_path(r"C:\PATH\TO\Simulation.factoryio")
5 app.web_server = True
6 scene.start_in_run_mode = True

```

Codeausschnitt 9.4: Erstellung des auto.cfg Files

Anhand der Einstellungen aus dem Codeausschnitt 9.4 sowie der benötigten Informationen in Zeile 1 und 3 öffnet Factory I/O automatisch die bereitgestellte Fertigungslinie und startet direkt im Startmodus. Weiters wird die Web API¹⁸ von Factory I/O gestartet und läuft auf Port 7410. Insofern keine Lizenz für Factory I/O erworben wurde, besteht die Möglichkeit die Testversion auszuprobieren. Falls letzteres der Fall ist, muss die erste Zeile, in welcher die Angabe der Lizenznummer benötigt wird, gelöscht werden.

Sobald alle Einstellungen vorgenommen wurden, zeigt Codeausschnitt 9.5, wie die Implementierung gestartet werden kann. Wenn alle notwendigen Dateien erstellt wurden, ist es ebenso möglich, die Implementierung mit einem einzigen Befehl zu starten, insofern der *MODBUS_SERVER* im .env-File aktualisiert wurde (siehe Codeausschnitt 9.6). Abbildung 9.13 zeigt wie Docker-Desktop nach einem erfolgreichen Start aussieht.

¹⁸<https://docs.factoryio.com/manual/web-api/>

9. Technische Implementierung

Tabelle 9.1 zeigt sämtliche Services, welche über ein eigenes User Interface (UI) verfügen und wo jene zu finden sind.

```
1 docker - compose up -- build -d
2 cd factoryio/sdk
3 dotnet run
```

Codeausschnitt 9.5: SKD von Factory I/O starten (Terminal)

```
1 # MODBUS_SERVER im .env-File aktualisieren!
2 cmd /k setup.bat
```

Codeausschnitt 9.6: Implementierung mit einem Befehl starten (Terminal)

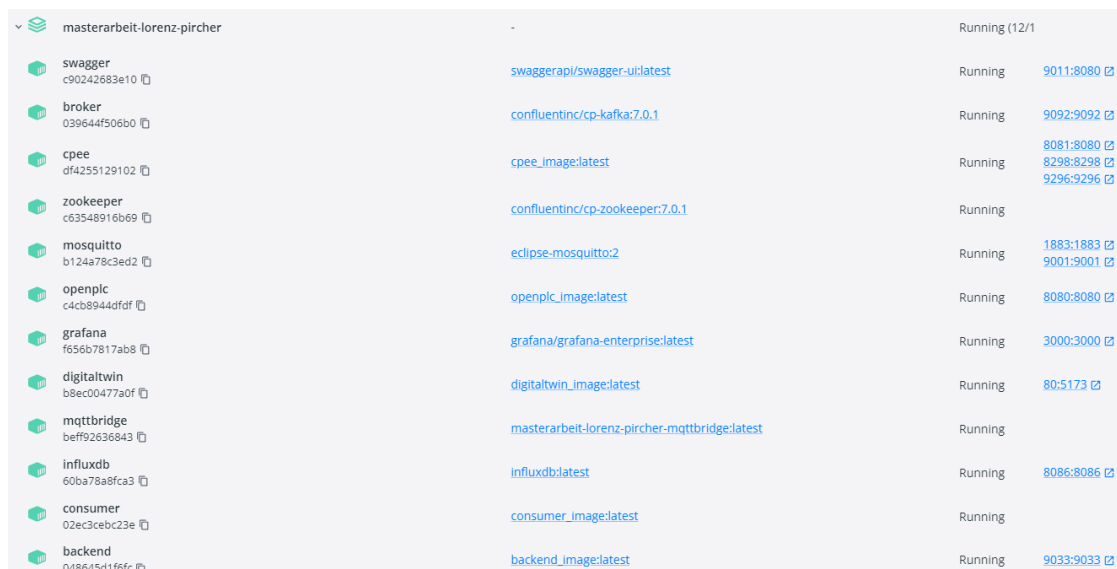


Abbildung 9.13.: Docker-Desktop nach dem Start der Implementierung (Screenshot)

Service	UI URL
TWAIN	http://localhost
Grafana	http://localhost:3000
OpenPLC	http://localhost:8080
CPEE	http://localhost:8081
InfluxDB	http://localhost:8086
Swagger	http://localhost:9011

Tabelle 9.1.: Microservice UI URL-Adressen

9.4.3. Simulations-Logs in der Konsole

Während der Simulation werden alle Ereignisse in der Konsole protokolliert, um zu verstehen, wann bestimmte Aufgaben ausgeführt wurden und ob diese erfolgreich waren.

Abbildung 9.14 zeigt die Konsole während der Ausführung der Simulation. Nachdem das SDK gestartet wurde und der MQTT-Producer läuft, kann über die TWAIN eine Simulation definiert oder eine bereits existierende ausgewählt und gestartet werden. Die dabei entstehenden Logs verfügen über eine zuordenbare ID sowie Task-Namen. Weiters enthalten diese einen Zeitstempel und einen Status. Sollten Fehler auftreten, werden diese ausgegeben.

```
Web Anwendung des digitalen Zwillings: <http://localhost/>
SDK wird gestartet ... (beliebige Taste zum Beenden, nicht Ctr + C)
SDK gestartet ...
MQTT Producer gestartet ...

Simulationen können gestartet werden ...
```

Task	Status	Zeitstempel	ID
Material auf Palette vorbereiten	Fertig	21.05.2023 11:49:50	-
RFID Transponder initialisieren	Fertig	21.05.2023 11:49:52	6174
Material zuschneiden mit Maschine B	Fertig	21.05.2023 11:50:56	6174
Loecher vorbohren (Schränk)	Fertig	21.05.2023 11:51:03	6174
Regale herstellen	Fertig	21.05.2023 11:51:09	6174
Spiegel zur Bearbeitung vorbereiten	Fertig	21.05.2023 11:51:18	6174
Standardtuer herstellen	Fertig	21.05.2023 11:51:26	6174
Spiegel rund zuschneiden	Fertig	21.05.2023 11:51:54	6174
Tuerschloss einbauen	Fertig	21.05.2023 11:51:56	6174
Spiegel auf Tuer befestigen	Fertig	21.05.2023 11:52:35	6174
Kleidenhaken und Schrankstange beilegen	Fertig	21.05.2023 11:52:41	6174
Oberflächenbehandlung	Fertig	21.05.2023 11:54:48	6174
Qualitätskontrolle	Fertig	21.05.2023 11:54:59	6174
Fehlende Bestandteile beilegen	Fertig	21.05.2023 11:55:16	6174
Qualitätskontrolle	Fertig	21.05.2023 11:55:38	6174
Zusatzausruestung beilegen	Fertig	21.05.2023 11:55:57	6174
In Box L verpacken	Fertig	21.05.2023 11:56:13	6174
Pakethoehe messen	Fertig	21.05.2023 11:56:15	6174
Paket wiegen	Fertig	21.05.2023 11:56:17	6174
Lieferschein fuer großes Paket drucken	Fertig	21.05.2023 11:56:28	6174
Material auf Palette vorbereiten	Fertig	21.05.2023 11:57:03	-
RFID Transponder initialisieren	Fertig	21.05.2023 11:57:05	6179
Material auf Palette vorbereiten	Fertig	21.05.2023 11:57:59	-
RFID Transponder initialisieren	Fertig	21.05.2023 11:58:01	6181
Material zuschneiden mit Maschine B	Fertig	21.05.2023 11:58:04	6179
Loecher vorbohren (Schreibtisch)	Fertig	21.05.2023 11:58:09	6179
Schubladen herstellen	Fertig	21.05.2023 11:58:13	6179
Tischplatte gravieren	Fertig	21.05.2023 11:58:18	6179
Tischbeine herstellen	Fertig	21.05.2023 11:58:22	6179

Abbildung 9.14.: Simulations-Logs in der Konsole (Screenshot)

9.5. Einblicke in den Code

Dieses Unterkapitel gibt Einblicke in die Implementierung von Tasks sowohl in C# als auch in NodeJS. Es demonstriert, wie Tasks über eine API gestartet werden können und dabei die Aktivierung von Maschinen und Geräten in Factory I/O erfolgt. Darüber hinaus wird veranschaulicht, wie Informationen aus Kafka Topics in einer Echtzeitdatenbank gespeichert und die Tasks des WfMS nach deren Abschluss weitergeschaltet werden.

9. Technische Implementierung

Codeausschnitt 9.7 zeigt die C# Klasse *RfidDevice*. Im SDK werden Objekte dieser Klasse instanziiert, um die Referenznummer des RFID-Transponders oder Werte an einer vorgegebenen Stelle zu lesen bzw. zu schreiben. Weiters wird eine Methode definiert, um zu prüfen, ob ein RFID-Transponder erkannt wurde. Weiters können die Details einer Bestellung bzw. Simulation abgerufen sowie Tasks abgeschlossen werden.

```
1 // ...
2 class Program
3 {
4     // ...
5     public static async Task<bool> completeTask(int rfidRef, string
6     endpoint, string payload)
7     {
8         // Methode zum Abschließen von Tasks
9     }
10    public class RfidDevice
11    {
12        private int amount;
13        private MemoryInt command, readRfid, writeRfid, memory_index,
14        command_id, status;
15        private MemoryBit rfid;
16
17        public RfidDevice(string rfidName)
18        {
19            // Konstruktor
20        }
21        public async Task<bool> foundTag(int delay)
22        {
23            // Kontrolliert, ob RFID Transponder erkannt wurde.
24        }
25        public async Task<int> readReference(int delay = 100, bool
26        checkTag = true)
27        {
28            // Liest die Referenznummer.
29        }
30        public async Task<int> readValue(int index, bool checkTag = true,
31        int delay = 100)
32        {
33            // Liest Wert an einen vorgegebenen Index.
34        }
35        public async Task writeValue(int index, int writeVal, int delay =
36        100)
37        {
38            // Schreibt einen Wert an einen vorgegebenen Index.
39        }
40        public async Task<Order> getOrder(int rfidRef)
41        {
42            // Ladet die Simulations- bzw. Bestelldetails.
```

```

43     // ...
44     RfidDevice rfidDevice = new RfidDevice("pre_drill_a_rfid");
45     int palletReference = await rfidDevice.readValue(0);
46     MemoryBit simulateLight = MemoryMap.Instance.GetBit("
simulate_pre_drill", MemoryType.Output);
47     simulateLight.Value = false;
48     bool taskSucceed = await completeTask(palletReference);
49     Console.WriteLine(FormatEvent("Vorbohren | " + (taskSucceed ? "
Fertig" : "Fehler") + " | " + DateTime.Now.ToString() + " | " +
palletReference));
50     }
51 }

```

Codeausschnitt 9.7: RFID-Leser Klasse (C#)

Codeausschnitt 9.8 zeigt wie Echtzeitdaten via MQTT weitergegeben werden. Hierzu wird eine statische Methode definiert, welche Werte verschiedener Datentypen verarbeiten kann.

```

1 class Program
2 {
3     // ...
4     public static async Task sendTopic(string deviceId, object
deviceValue)
5     {
6         var mqttMsg = new JObject();
7         if (deviceValue is string)
8         {
9             mqttMsg["val"] = deviceValue as string;
10        }
11        else if (deviceValue is int)
12        {
13            mqttMsg["val"] = (int)deviceValue;
14        }
15        else if (deviceValue is bool)
16        {
17            mqttMsg["val"] = (bool)deviceValue;
18        }
19        else if (deviceValue is float)
20        {
21            mqttMsg["val"] = (float)deviceValue;
22        }
23        mqttMsg["timestamp"] = DateTime.Now;
24        string payload = JsonConvert.SerializeObject(mqttMsg);
25        var mqttMessage = new MqttApplicationMessageBuilder()
26            .WithTopic($"device/{deviceId}")
27            .WithPayload(payload)
28            .Build();
29        await _client.PublishAsync(mqttMessage);
30    }
31
32    // ...

```

Codeausschnitt 9.8: MQTT Topic via SDK senden (C#)

9. Technische Implementierung

Der Codeausschnitt 9.9 veranschaulicht, wie die Simulations-API Tasks behandelt, die vom WfMS initiiert wurden. Hierzu werden Informationen wie der Task-Name sowie die WfMS-Instanz und Callback-URL in einem Objekt gespeichert. Dabei wird insbesondere berücksichtigt, um welche Art von Task es sich handelt. Die Methode *updatePart* erlaubt außerdem das Abändern von Werten in Factory I/O über die verfügbare API. Schließlich wird im Codeausschnitt dargestellt, wie beispielsweise eine Maschine gestartet wird. Hierzu werden zunächst die oben erwähnten Informationen im Simulationsobjekt gespeichert. Anschließend wird die Maschine in Factory I/O eingeschaltet. Durch das Setzen des Headers *cpee-callback = true* wartet das WfMS auf die Abschlussmeldung des Tasks in Factory I/O mittels eines Callbacks.

```
1 // Imports
2 // ...
3 async function registerTask(req, reference = null, palletTask = false) {
4   const callback = req.headers["cpee-callback"].replace("https", "http")
5   ).replace("localhost", "host.docker.internal");
6   const taskName = req.headers["cpee-label"];
7   if (!reference) {
8     const instance = req.headers["cpee-instance"];
9     return {
10       callback: callback,
11       taskName: taskName,
12       instance: instance
13     };
14   }
15   let theOrder = map.get(reference);
16   if (!palletTask) {
17     theOrder.status = "In Progress";
18     theOrder.callback = callback;
19     theOrder.currentTask = taskName;
20     await writeJson(theOrder);
21   } else {
22     theOrder.palletCallback = callback;
23     theOrder.currentPalletTask = taskName;
24     theOrder.palletStatus = "In Progress";
25     await writeJson(theOrder, true, true);
26   }
27   return theOrder;
28 }
29 async function updatePart(partName, value) {
30   const body_on = [
31     {
32       name: partName,
33       value: value
34     }
35   ];
36   await fetch("http://host.docker.internal:7410/api/tag/values/by-name", {
37     method: "PUT",
38     headers: {
39       "Content-Type": "application/json"
```

```

40     body: JSON.stringify(body_on)
41   });
42 }
43 // ...
44 router.post("/machining", async (req, res) => {
45   let theOrder = await registerTask(req, req.body.reference);
46   theOrder.direction = req.body.machine === "A" ? 1 : 0;
47   if (req.body.machine === "A") {
48     await updatePart("machining_A_start", true);
49   } else {
50     await updatePart("machining_B_start", true);
51   }
52   map.set(req.body.reference, theOrder);
53   res.status(200).setHeader("cpee-callback", true).send();
54 });

```

Codeausschnitt 9.9: Task einleiten (NodeJS)

Im Codeausschnitt 9.10 wird gezeigt, wie das WfMS benachrichtigt wird, nachdem ein Task in der Simulationsumgebung abgeschlossen wurde. Hierzu wird eine Methode definiert, die einen HTTP PUT-Request an das WfMS sendet. Standardmäßig übergibt die Methode keine Daten. Es besteht optional die Möglichkeit, Daten an das WfMS zu übermitteln. Schließlich zeigt der Codeausschnitt einen Endpoint, der für den Abschluss der meisten Tasks im SDK verwendet wird.

```

1  // Imports
2  // ...
3  async function sendCallback(callback, jsonBody = null) {
4    if (!jsonBody) {
5      await fetch(callback, {
6        method: "PUT",
7      });
8    } else {
9      await fetch(callback, {
10       method: "PUT",
11       headers: {
12         "Content-Type": "application/json"
13       },
14       body: JSON.stringify(jsonBody)
15     });
16   }
17 }
18 // ...
19 router.post("/complete", async (req, res) => {
20   let theOrder = map.get(req.body.reference);
21   theOrder.status = "Completed";
22   await writeJson(theOrder, false);
23   map.set(req.body.reference, theOrder);
24   await sendCallback(theOrder.callback);
25   res.status(200).send();
26 });
27 // ...

```

Codeausschnitt 9.10: Task beenden (NodeJS)

9. Technische Implementierung

Codeausschnitt 9.11 zeigt den Kafka Consumer, der eingehende Topics in einer Echtzeitdatenbank speichert. Zunächst werden die Topics aufgelistet, die abonniert werden sollen. Wenn eine Nachricht das entsprechende Topic aufweist, wird der entsprechende *Point* erstellt und in InfluxDB gespeichert.

```
1 // Imports
2 // ...
3 const consumer = kafka.consumer({ groupId: "factory-data-stream" });
4
5 const run = async () => {
6   await consumer.connect();
7   await consumer.subscribe({
8     topics: ["device-startsensor"], // gekuerzt
9     fromBeginning: true
10  });
11  await consumer.run({
12    eachMessage: async ({ topic, partition, message }) => {
13      // ...
14      const deviceid = topic.split("-")[1]
15      const payload = JSON.parse(message.value.toString());
16      let point;
17      if (deviceid === "startsensor") {
18        point = new Point("sensor")
19          .tag("name", "startsensor")
20          .intField("val", payload.val)
21          .timestamp(new Date(payload.timestamp));
22      }
23      // ...
24      writeApi
25        .writePoint(point);
26      try {
27        await writeApi.flush()
28      } catch (e) {
29        console.error()
30      }
31    }
32  });
33 };
34 run().catch(e => console.error(`[factory/consumer] ${e.message}`, e));
```

Codeausschnitt 9.11: Kafka Topic in InfluxDB speichern (NodeJS)

9.6. API Beschreibung

Abbildung 9.15 zeigt die Beschreibung der Simulations-API. Diese API wird sowohl vom digitalen Zwilling, größtenteils über das SDK von Factory I/O, als auch vom WfMS verwendet. Die Endpoints in der Beschreibung lassen sich in verschiedene Kategorien einteilen. Einige dienen dazu, Aufgaben sowohl in der Simulationumgebung als auch im WfMS für beide Produktarten auszulösen, während andere spezifisch für ein bestimmtes Produkt herangezogen werden können. Folglich gibt es auch andere Services, die zum

Abschließen von Aufgaben dienen. Zusätzlich ermöglichen weitere Endpoints das Abrufen von Simulationsdetails und das Hinterlegen und Abrufen von Referenznummern. Des Weiteren stehen Services zur Verfügung, die die Verwaltung und Erstellung von WfMS-Instanzen ermöglichen. Die letzte Kategorie dient der Healthcheck-Unterseite der TWAIN.

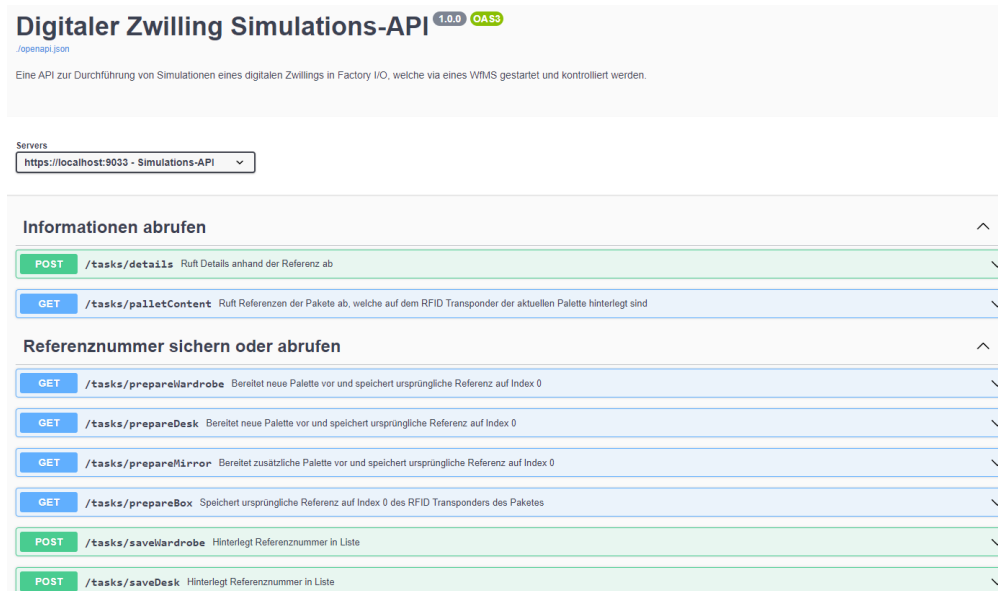


Abbildung 9.15.: Simulations-API Beschreibung in Swagger (Screenshot)

9.7. Limitierungen

Der verwendete Treiber in Factory I/O, Modbus TCP/IP Server, ermöglicht die Nutzung von je 256 digitalen Eingängen sowie Ausgängen. Zusätzlich können je 64 Register Ein- sowie Ausgänge kontrolliert werden [115]. Jene Begrenzung schränkt die Anzahl der möglichen Sensoren, Geräte und Maschinen ein, welche anhand des PLC kontrolliert bzw. automatisiert werden sollen. Für das SDK sowie die Web API von Factory I/O werden keine maximalen Speicherindizes erwähnt. Für die Implementierung beider Anwendungsfälle reicht jene Menge an Ein- und Ausgängen zur Genüge. Insofern der zugrunde liegende Prozess erweitert werden sollte und folglich nicht ausreichend Ein- und Ausgänge vorhanden sind, macht es Sinn, jenen Prozess in kleinere Unterprozesse aufzuteilen.

10. Anwendungsfälle

Im Rahmen dieser Arbeit werden zwei Anwendungsfälle vorgestellt, bei denen ein digitaler Zwilling einer Fertigungslinie von einem WfMS gesteuert wird. Jene beiden Anwendungsfälle implementieren zwei Prozesse, welche im folgenden Unterkapitel näher erläutert werden:

1. Produktion von Möbel
2. Verpackung und Versand

Zur Umsetzung dieser Prozesse wird eine Fertigungslinie in Factory I/O erstellt, welche schließlich eine 3D-Simulation ermöglicht. Der erste Prozess erlaubt die Herstellung von zwei verschiedenen Produkten. Hierbei regelt das WfMS unter anderem, welches Material für die Produktion vorbereitet sowie welche Maschinen oder Wege in der Fertigungslinie genommen werden sollen. Sobald die Produktion abgeschlossen ist, erfolgt der nächste Prozess, welcher logistische Arbeiten beinhaltet. Das WfMS ist dabei für die Wahl des Verpackungsmaterials sowie der passenden Lieferoption verantwortlich. Im Anschluss verlässt das Produkt schließlich die Fabrik. Während der gesamten Simulation wird der Ablauf von einem WfMS gesteuert, das sämtliche Entscheidungen übernimmt und somit bestimmt, wie der Prozess abgewickelt wird. Dies erfolgt durch das Lesen und Schreiben von RFID-Werten oder anderen Sensordaten und basiert auf vorher festgelegten Prozessangaben. Der Einsatz von RFID-Technologie und der WfMS-Steuerung ermöglicht eine effiziente, automatisierte und fehlerfreie Abwicklung der Prozesse.

Die Implementierung beider Anwendungsfälle soll skalierbar, zuverlässig, benutzerfreundlich und verständlich sein. Die Skalierbarkeit soll dadurch gewährleistet werden, indem der digitale Zwilling um eine große Menge an Maschinen, Sensoren und Geräten erweitert werden kann. Zudem sollen riesen Datenmengen in Echtzeit analysiert und angezeigt werden können. Zur Realisierung der Zuverlässigkeit soll anhand bewährter Methoden eine zuverlässige Kommunikation zwischen Sensoren, Maschinen und sonstigen Geräten und dem digitalen Zwilling aufgebaut werden, welche insbesondere für IoT-Geräte geeignet ist. Weiters sollen mit Hilfe einer verständlichen Notation Benutzer in der Lage sein, die Prozesse der Fertigungslinie zu verstehen und bei Bedarf anzupassen. Außerdem können eigene Simulationen über eine intuitive Web-Anwendung erstellt, durchgeführt und beobachtet werden.

10.1. Anwendungsfall 1: Produktion von Möbel

Im zugrundeliegenden Prozess von Anwendungsfall 1 gibt es zwei mögliche Wege, um entweder einen Schrank oder einen Schreibtisch herzustellen. Abhängig von dem vom

10. Anwendungsfälle

Kunden bestellten Produkt wird das benötigte Material vorbereitet. In Factory I/O unterscheidet sich das Material lediglich in der Farbe, was jedoch zur Visualisierung völlig ausreichend ist. Anschließend wird die Referenznummer des RFID Transponders, welche Factory I/O zufällig vergibt, als Schlüssel in einer Hashmap hinterlegt. Der Wert dieses Schlüssels ist ein JavaScript Object Notation (JSON)-Objekt, welches sämtliche Bestell- und WfMS-Details (z.B. Instanz- und Callback-Informationen) enthält. Wenn das WfMS Entscheidungen trifft, werden diese mithilfe eines RFID-Lesegerätes auf den RFID-Transponder geschrieben. Zum Beispiel wird auf Index 1 des RFID-Transponders hinterlegt, mit welcher Maschine das Material als erstes zugeschnitten werden soll, indem die Richtung angegeben wird, welche die Palette mit dem Material nehmen soll. Für einen Schreibtisch wird Maschine A verwendet und für einen Schrank Maschine B, was zu den zwei unterschiedlichen Prozesspfaden führt. Beide Prozesswege beginnen mit dem Vorbohren von Löchern. Im Prozessweg zur Herstellung eines Schreibtisches werden zunächst Schubladen hergestellt. Anschließend folgt die Entscheidung, ob die Tischplatte klassisch graviert oder eine Kabeldurchführung eingefügt werden soll, wobei die entsprechenden Informationen aus den Bestelldetails stammen. Danach werden die Tischbeine produziert. Beim Prozessweg zur Herstellung eines Schrankes werden nach dem Vorbohren der Löcher zunächst die Regale hergestellt. Anschließend erfolgt die Herstellung der gewünschten Türart und die Vorbereitung und Bearbeitung des Spiegels, welcher später auf der Tür befestigt wird. Die Form des Spiegels und die Türart werden in den Bestelldetails definiert. Schließlich wird noch ein Kleiderhaken befestigt und eine Schrankstange beigelegt. Danach münden beide Prozesswege wieder zusammen und es folgt eine Oberflächenbehandlung, in welcher das Produkt lackiert wird. Im Anschluss erfolgt eine iterative Qualitätskontrolle, bei der die Vollständigkeit der Bestellung geprüft wird, indem das WfMS einen Zufallswert generiert. Wenn das Produkt unvollständig ist, wird der fehlende Bestandteil beigelegt und es erfolgt eine erneute Qualitätskontrolle. Nach erfolgreicher Qualitätskontrolle endet der Prozess. Abbildung 10.1 veranschaulicht den Prozess von Anwendungsfall 1 in BPMN.

Zur Kommunikation zwischen den einzelnen Sensoren, Geräten sowie Maschinen der Simulation und dem digitalen Zwilling wird eine End-to-End Integration via MQTT und Kafka realisiert. Der Grund für jenen Ansatz ist, dass sich die Kombination jener beiden Technologien insbesondere für IoT-Geräte eignet, da diese zu einer erhöhten Skalierbarkeit sowie Zuverlässigkeit führen kann.

Der aktuelle Fortschritt der Produktion wird in der Workflow Engine angezeigt. Weiters können über die TWAIN Echtzeit-Daten im Dashboard eingesehen und bei Bedarf mit Hilfe von InfluxDB in Echtzeit analysiert werden.

Der in BPMN dargestellte Prozess der Produktion (siehe Abbildung 10.1) kann von sämtlichen Benutzern verstanden und ohne spezielles Fachwissen abgeändert werden.

10.1. Anwendungsfall 1: Produktion von Möbel

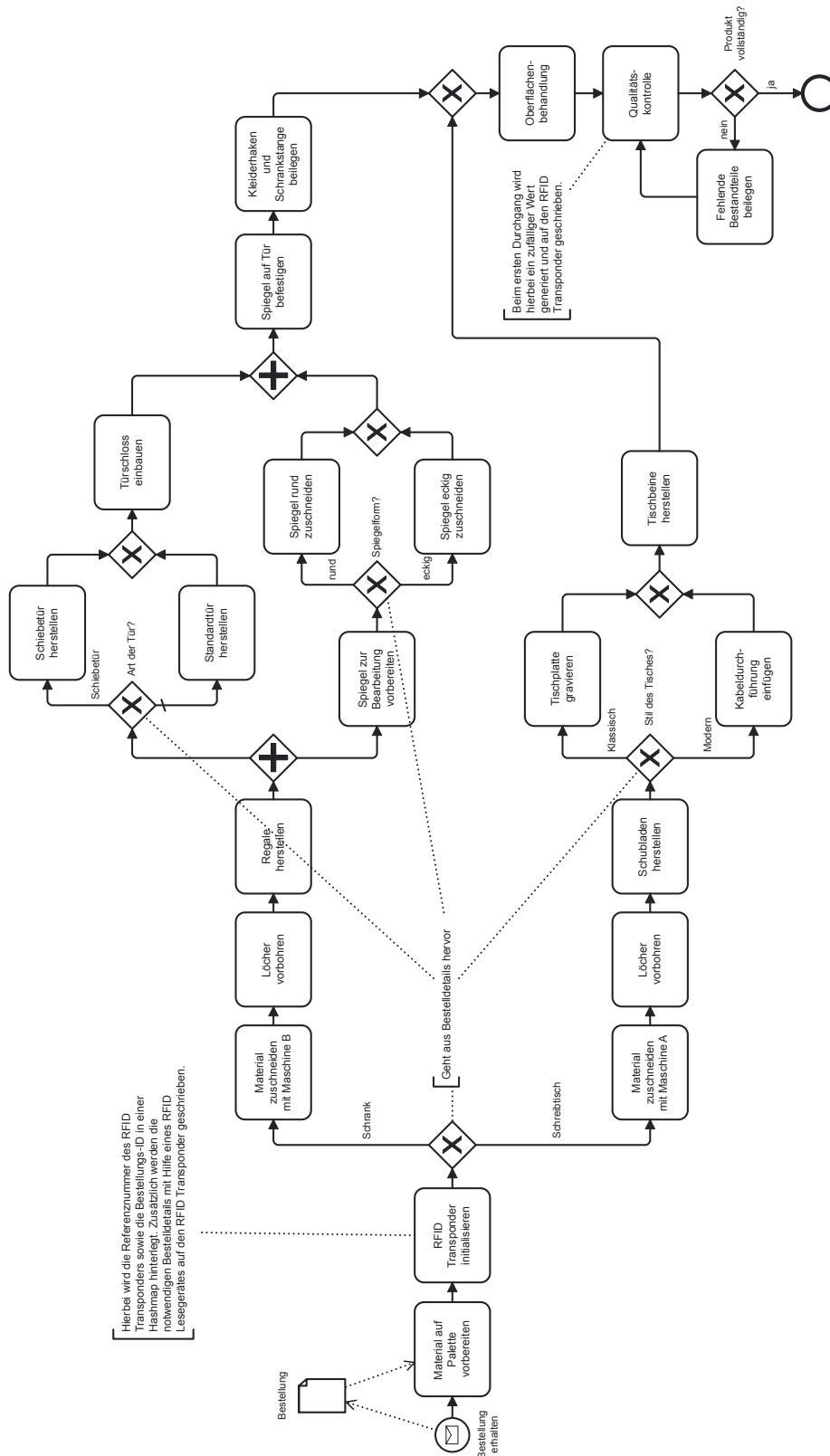


Abbildung 10.1.: Anwendungsfall 1: Produktion von Möbel (BPMN-Diagramm) 81

10. Anwendungsfälle

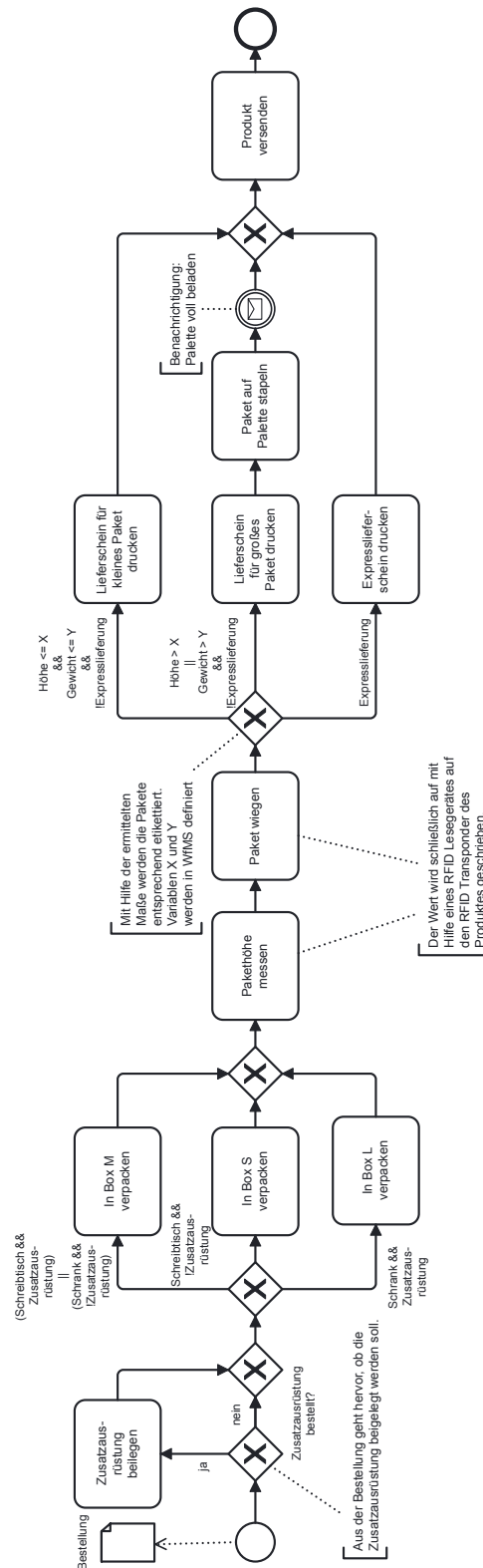


Abbildung 10.2.: Anwendungsfall 2: Verpackung und Versand (BPMN-Diagramm)

10.2. Anwendungsfall 2: Verpackung und Versand

Nach Abschluss der Produktion wird in der Simulation automatisch der Verpackungs- und Versandprozess, also Anwendungsfall 2, eingeleitet. Zunächst wird anhand der Bestelldetails überprüft, ob eine Zusatzausrüstung in Form eines Werkzeugkoffers beigelegt werden soll und entsprechend gehandelt. Anschließend wird anhand der Produktart sowie der Zusatzausrüstungsangabe (Boolean) entschieden, welches Verpackungsmaterial verwendet werden soll. Hierzu stehen in Factory I/O drei verschiedene Paketgrößen (S, M, L) zur Verfügung. Im nächsten Schritt wird die Höhe des Pakets mittels eines Sensors bestimmt und auf dem RFID-Transponder hinterlegt. Ebenso wird das Gewicht des Pakets mittels einer Waage ermittelt und auf den nächsten freien Speicherindex geschrieben. Anhand jener Werte sowie der Express-Angabe (Boolean) aus den Bestelldetails geht hervor, ob das Produkt sofort versendet oder ob dieses zunächst auf eine Palette geladen und erst später, sobald diese voll ist, verschickt werden soll. Der dargestellte BPMN-Prozess in Abbildung 10.2 bezieht sich auf Anwendungsfall 2 und somit auf das Verpacken und Versenden sämtlicher Produkte in der Fertigungslinie.

Auch in diesem Anwendungsfall wird zur Kommunikation zwischen den einzelnen Sensoren, Geräten sowie Maschinen der Simulation und dem digitalen Zwilling eine End-to-End Integration via MQTT und Kafka realisiert, welche sich aufgrund der hohen Skalierbarkeit sowie Zuverlässigkeit insbesondere für IoT-Geräte eignet.

Der aktuelle Fortschritt der Produktion wird in der Workflow Engine angezeigt. Weiters können in einer benutzerfreundlichen TWAIN Echtzeit-Daten im Dashboard eingesehen und bei Bedarf mit Hilfe von InfluxDB in Echtzeit analysiert werden.

Der in BPMN dargestellte Verpackungs- und Versandprozess (siehe Abbildung 10.2) kann von sämtlichen Benutzern verstanden und ohne spezielles Fachwissen abgeändert werden.

10.3. Überprüfung der WF-Net Eigenschaften

Im nächsten Schritt wird eine Prozessverifikation durchgeführt, die auf der Theorie aus Kapitel 5.7 basiert. Das Ziel dieser Verifikation ist es, sicherzustellen, dass das Modell syntaktisch korrekt ist, sowohl strukturell als auch verhaltensbezogen. Hierzu wird unter anderem untersucht, ob das WF-Net strukturell solide, lebendig sowie begrenzt ist. Dabei werden lediglich formale Eigenschaften des Prozessmodells beachtet, ohne den realen Prozess zu kennen. Im vorliegenden Fall werden beide BPMN-Modelle zunächst als Petri-Netz, genauer gesagt als WF-Netz, abgebildet, damit jene schließlich einer Prozessverifikation unterzogen werden können. Jene beiden Abbildungen 10.3 und 10.4 enthalten dementsprechend keine formal irrelevanten Informationen wie beispielsweise Stellen- oder Transitionsbeschreibungen. Aus Gründen der Vollständigkeit werden diese dennoch im Anhang in den Tabellen A.2 sowie A.3 aufgeführt.

10. Anwendungsfälle

Produktion von Möbel

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}, p_{27}, p_{28}, p_{29}, p_{30}, p_{31}, p_{32}, p_{33}, p_{34}, p_{35}, p_{36}, p_{37}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}, t_{31}\}$
- $F = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (t_2, p_3), (p_3, t_3), (t_3, p_4), (p_4, t_4), (t_4, p_5), (p_5, t_5), (t_5, p_6), (p_6, t_6), (t_6, p_7), (p_7, t_7), (t_7, p_8), (p_8, t_8), (t_8, p_9), (p_9, t_9), (t_9, p_{10}), (p_{10}, t_{11}), (t_{11}, p_{11}), (p_{11}, t_{10}), (t_{10}, p_{12}), (p_{12}, t_{11}), (t_{11}, p_{13}), (p_{13}, t_{12}), (t_{12}, p_{14}), (p_{14}, t_{19}), (t_{19}, p_{15}), (p_{15}, t_{14}), (t_{14}, p_{16}), (p_{16}, t_{15}), (t_{15}, p_{17}), (p_{17}, t_{16}), (t_{16}, p_{18}), (p_{18}, t_{18}), (t_{18}, p_{19}), (p_{19}, t_{17}), (t_{17}, p_{20}), (p_{20}, t_{18}), (t_{18}, p_{21}), (p_{21}, t_{19}), (t_{19}, p_{22}), (p_{22}, t_{20}), (t_{20}, p_{23}), (p_{23}, t_{27}), (t_{27}, p_{24}), (p_{24}, t_{19}), (t_{19}, p_{25}), (p_{25}, t_{20}), (t_{20}, p_{26}), (p_{26}, t_{21}), (t_{21}, p_{27}), (p_{27}, t_{22}), (t_{22}, p_{28}), (p_{28}, t_{23}), (t_{23}, p_{29}), (p_{29}, t_{25}), (t_{25}, p_{30}), (p_{30}, t_{24}), (t_{24}, p_{31}), (p_{31}, t_{25}), (t_{25}, p_{32}), (p_{32}, t_{26}), (t_{26}, p_{33}), (p_{33}, t_{27}), (t_{27}, p_{34}), (p_{34}, t_{28}), (t_{28}, p_{35}), (p_{35}, t_{29}), (t_{29}, p_{36}), (p_{36}, t_{30}), (t_{30}, p_{35}), (p_{36}, t_{31}), (t_{31}, p_{37})\}$

Verpackung und Versand

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}\}$
- $F = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (t_2, p_3), (p_3, t_3), (t_3, p_4), (p_4, t_3), (t_3, p_5), (p_5, t_4), (t_4, p_6), (p_6, t_5), (t_5, p_7), (p_7, t_8), (t_8, p_8), (p_8, t_6), (t_6, p_9), (p_9, t_8), (t_8, p_{10}), (p_{10}, t_7), (t_7, p_{11}), (p_{11}, t_8), (t_8, p_{12}), (p_{12}, t_9), (p_{13}, t_{10}), (t_{10}, p_{14}), (p_{14}, t_{11}), (t_{11}, p_{15}), (p_{15}, t_{12}), (t_{12}, p_{16}), (p_{16}, t_{17}), (t_{17}, p_{18}), (p_{18}, t_{14}), (t_{14}, p_{19}), (p_{19}, t_{15}), (t_{15}, p_{20}), (p_{20}, t_{17}), (t_{17}, p_{21}), (p_{21}, t_{16}), (t_{16}, p_{22}), (p_{22}, t_{17}), (t_{17}, p_{23}), (p_{23}, t_{18}), (t_{18}, p_{24})\}$

Beide Petri-Netze erfüllen alle Eigenschaften eines WF-Nets und sind strukturell solide. Aus der Strukturanalyse geht hervor, dass es jeweils genau eine Start- (Quelle) und eine Endstelle (Senke) gibt. Weiters haben alle Kanten ein zulässiges Gewicht (standardmäßig beträgt dieses eins). Ebenso ist die Beschränktheit erfüllt, da sich in keiner Stelle unendlich viele Markierungen ansammeln können. Da auch keine toten oder nicht lebendigen Transitionen vorliegen, ist auch die Lebendigkeit gegeben.

10.3. Überprüfung der WF-Net Eigenschaften

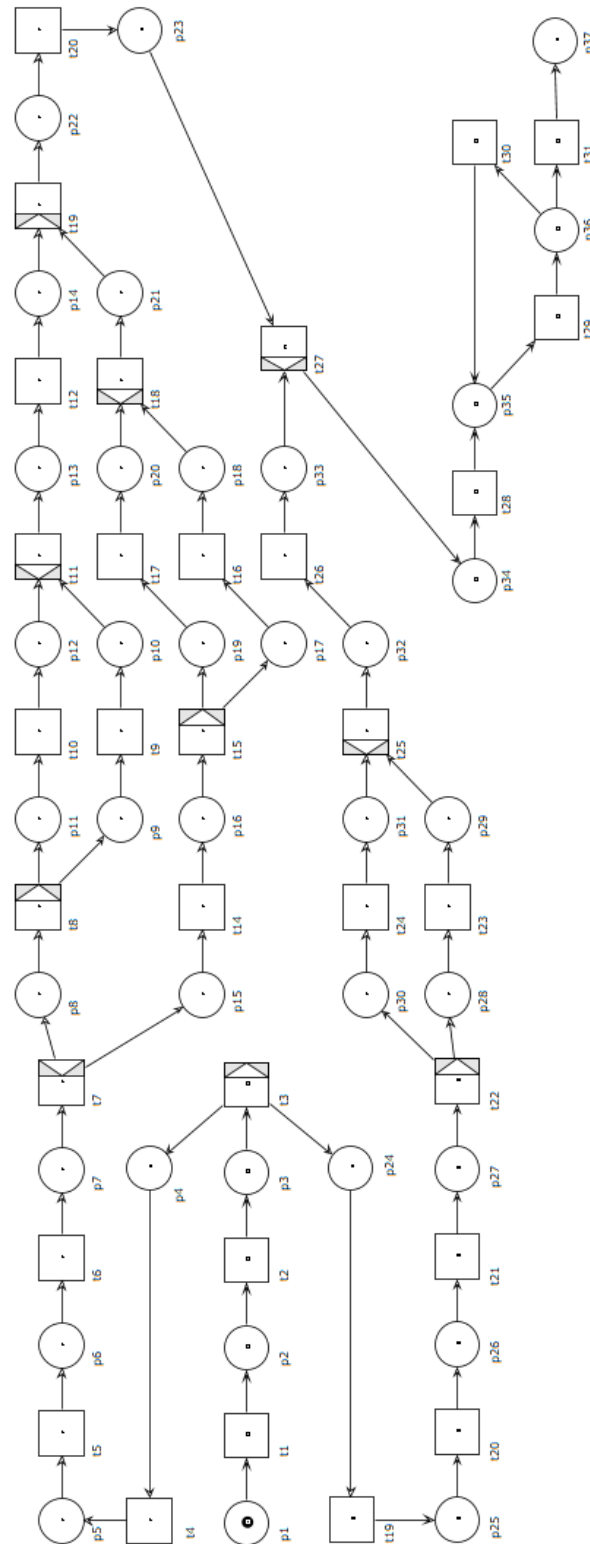


Abbildung 10.3.: Anwendungsfall 1: Produktion von Möbel als WF-Net

10. Anwendungsfälle

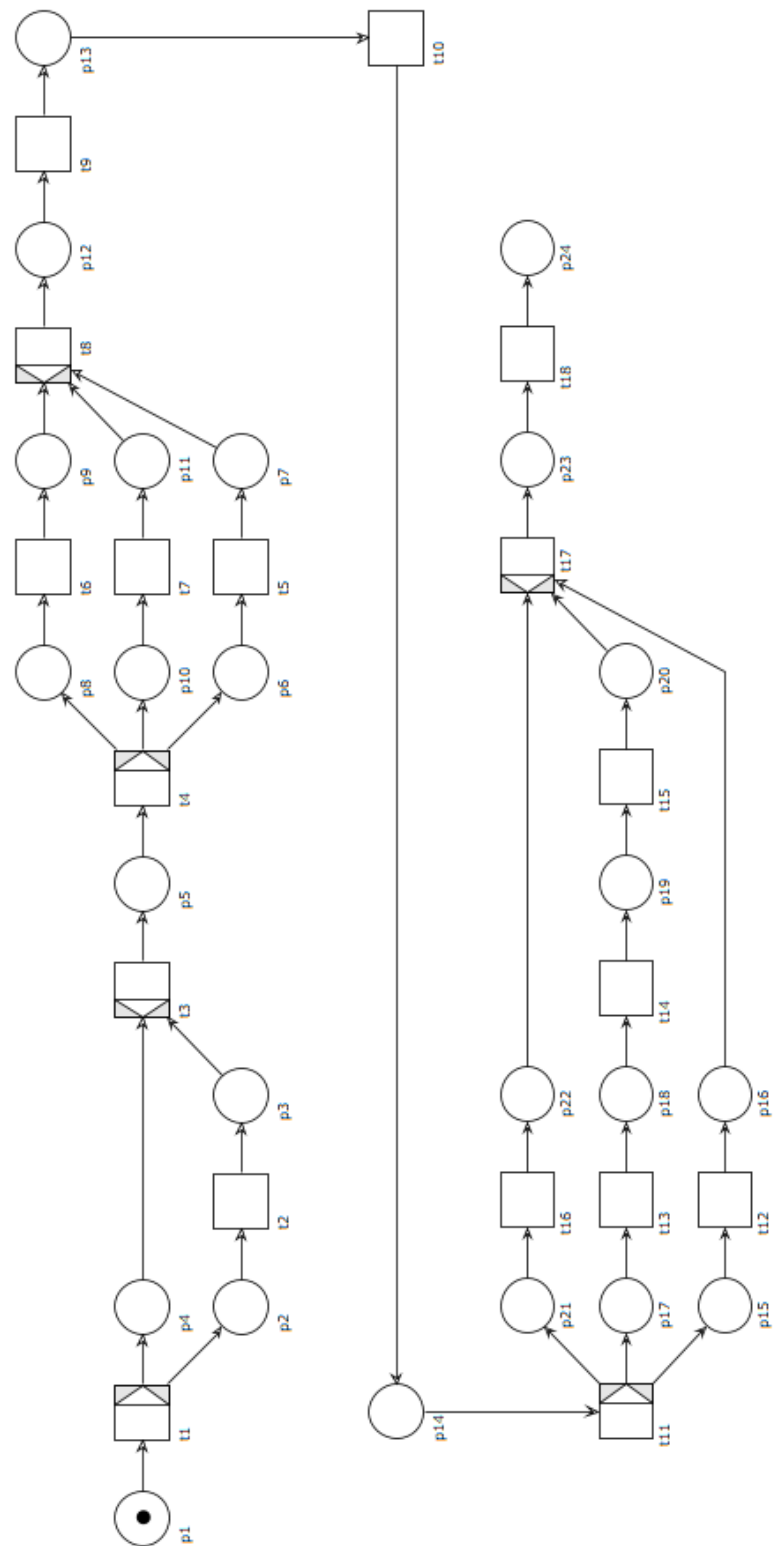


Abbildung 10.4.: Anwendungsfall 2: Verpackung und Versand als WF-Net

11. Evaluierung

In diesem Kapitel werden die Anwendungsfälle, der Implementierungsansatz sowie die Schnittstelle für die Nutzer über die TWAIN untersucht und bewertet. Dabei wird ein genauer Blick auf die Umsetzung und Ausführung beider Anwendungsfälle des digitalen Zwillings in der Simulationsumgebung sowie auf das Konzept und entscheidende Technologien geworfen.

Beide Anwendungsfälle kombinieren eine Reihe von typischen Anwendungen digitaler Zwillinge in Fabriken nach Köhler [54]. Durch die Anlagenplanung in Form einer virtuellen Abbildung in Factory I/O können Prozessabläufe der geplanten Anlage optimiert sowie Fehler in der Planung und Konstruktion frühzeitig erkannt werden, noch bevor die Anlage tatsächlich gebaut wird. So können zahlreiche Aspekte vorzeitig überprüft sowie eine virtuelle Umgebung, in der die Anlage begebar, erlebbar und interaktiv bedienbar ist, getestet werden. Durch die virtuelle Inbetriebnahme können Fehler reduziert sowie Routinen und Checks vorverlegt werden. Des Weiteren kann die Produktionslogistik vor dem tatsächlichen Produktionsstart simuliert werden, bei welcher sämtliche Abläufe ausgehend vom Rohmaterialien bis zum Endprodukt analysiert und optimiert werden können. Da dem digitalen Zwilling während des gesamten Betriebs der Fertigungslinie alle Daten der Simulation zur Verfügung stehen, ist es möglich, den Prozess zu steuern, zu überwachen und Optimierungspotenziale zu erkennen. Außerdem ermöglicht der digitale Zwilling die Simulation von Produktionsumstellungen, da dieser auf Prozesse basiert, welche stets von Nutzern angepasst werden können, umso deren Funktionalität und Effizienz zu überprüfen [54].

Im Rahmen dieser Arbeit konnten beide Anwendungsfälle implementiert werden, indem zunächst die benötigten Sensoren sowie Geräte mittels Factory I/O in einer virtuellen Fertigungslinie eingerichtet und anschließend mit einem PLC automatisiert wurden. Weiters wurden die Prozesse als Workflow abgebildet, um folglich vom WfMS ausgeführt werden zu können. Hierfür wurden unter anderem eine REST API, mit zahlreichen Endpoints, sowie weitere Microservices erstellt. Weiters können Nutzer ohne erforderliche Expertise über eine intuitive Web-Anwendung Echtzeitwerte des Zwillings überwachen sowie Simulationen erstellen, bearbeiten und ausführen. Diese in Kombination mit der Web API von Factory I/O ermöglicht das Einleiten einzelner Prozess-Tasks. Mit Hilfe des SDK werden Daten auf den RFID Transponder geschrieben oder gelesen sowie Tasks des WfMS abgeschlossen. Durch MQTT und Kafka wurde eine skalierbare und zuverlässige End-to-End Integration implementiert, wodurch Daten von Maschinen und Sensoren von der Echtzeitdatenbank InfluxDB erfasst und schließlich via Grafana in Form eines Dashboards abgerufen werden können.

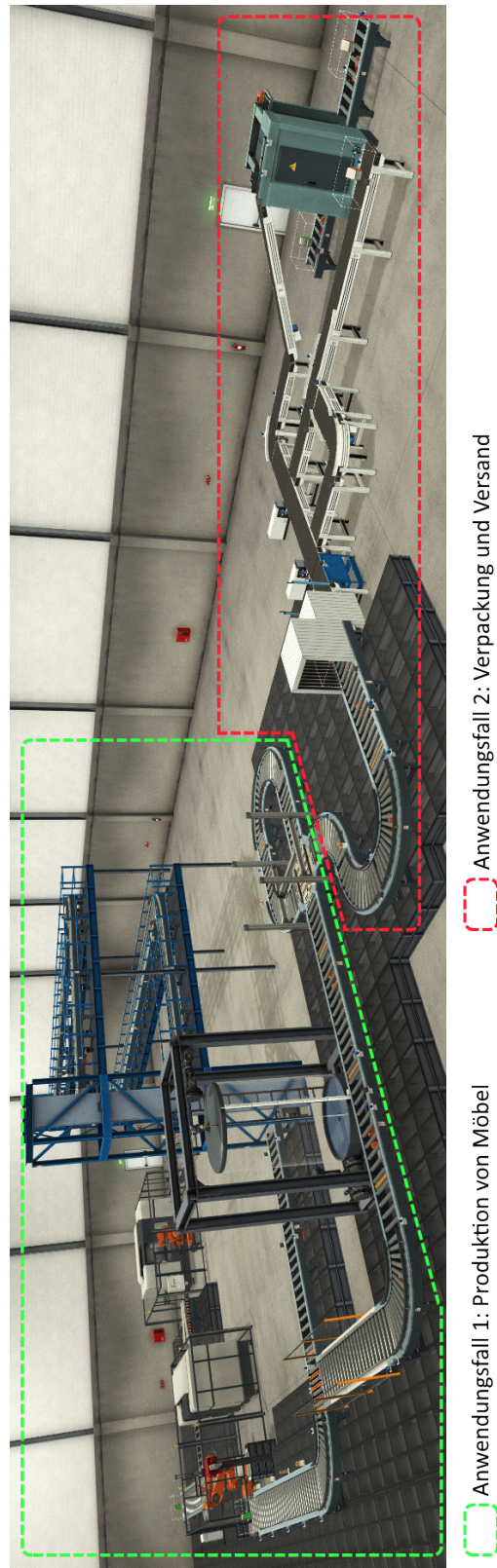


Abbildung 11.1.: Überblick: Gesamte Fertigungslinie in Factory I/O (Screenshot)

Abbildung 11.1 gibt einen umfassenden Überblick über die gesamte Fertigungslinie. Diese besteht aus zwei Teilen, von denen jeder für einen spezifischen Anwendungsfall verantwortlich ist. Zuerst wird Anwendungsfall 1: Produktion von Möbeln abgeschlossen, gefolgt von Anwendungsfall 2: Verpackung und Versand. In der Simulationssoftware sind daher beide Anwendungsfälle hintereinander auf ein und derselben Fertigungslinie abgebildet. Die virtuelle Fertigungslinie verfügt über mehr als 450 Tags, darunter Sensoren, Maschinen und andere Geräte, wovon ein Großteil entweder über den PLC oder das SDK kontrolliert wird. Im Anhang enthält Tabelle A.1 Informationen zu allen verwendeten Tags in Factory I/O. Sie zeigt den Namen, den Datentyp und ob es sich um einen Eingang oder Ausgang handelt. Darüber hinaus ist ersichtlich, für welchen Task der Tag benötigt wird und ob er eine Adresse belegt, welche vom PLC überwacht und kontrolliert wird.

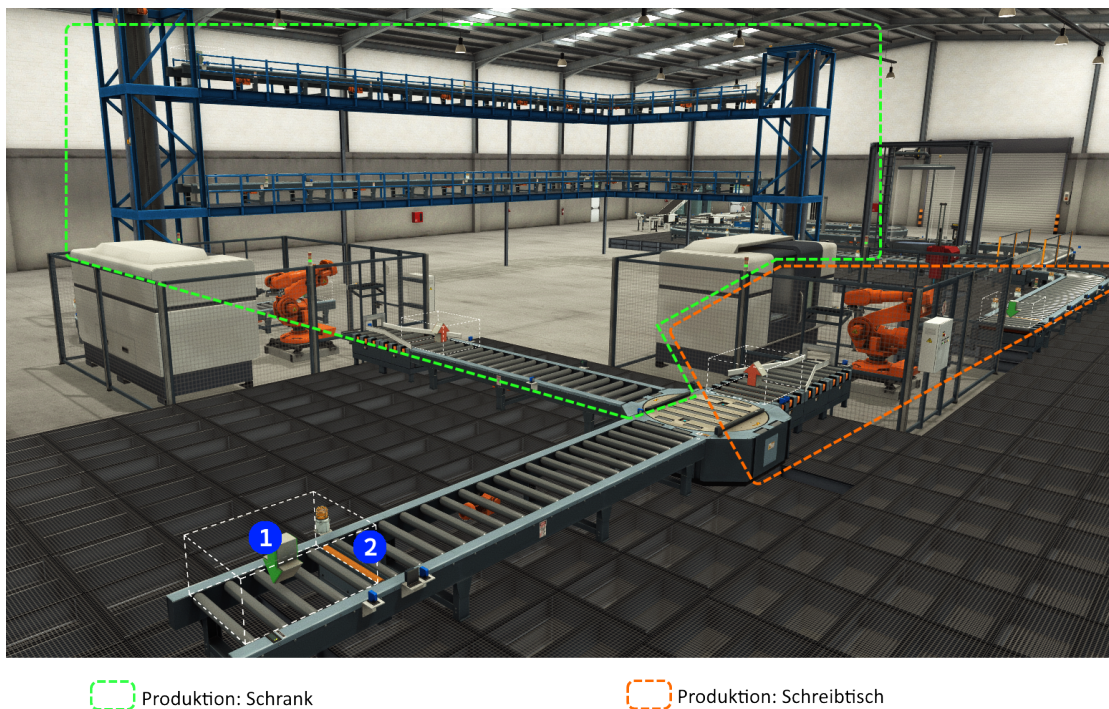


Abbildung 11.2.: Startpunkt der Fertigungslinie in Factory I/O (Screenshot)

Abbildung 11.2 zeigt den Startpunkt der Fertigungslinie, welcher zunächst Anwendungsfall 1: Produktion von Möbel einleitet.

① zeigt einen *Emitter*, welcher beliebige Objekte in Factory I/O hervorbringen kann. Dabei wird je nach erforderlichem Produkt ein unterschiedliches Material, ersichtlich durch verschiedene Farben auf einer Palette, welche über einen RFID-Transponder verfügt, vorbereitet. Sobald ein Sensor erkennt, dass ein neues Objekt auf dem Fließband platziert wurde, wird via MQTT ein Ereignis gesendet, was darüber informiert, dass eine Simulation

11. Evaluierung

gestartet wurde. Weiters verständigt das SDK das WfMS über den erfolgreichen Abschluss des Tasks, welches schließlich den nächsten einleitet.

② kümmert sich um die Initialisierung der Bestellung bzw. Simulation. Hierzu werden die Referenznummer des RFID-Transponders hinterlegt sowie anhand eines RFID-Lesers relevante Informationen auf diesen geschrieben. Schließlich wird basierend auf der Produktart entschieden, mit welcher Maschine das Material zugeschnitten werden soll. Demzufolge ergeben sich daraus zwei Prozesswege, wovon einer für die Produktion von Schreibtischen und ein weiterer für die Produktion von Schränken zuständig ist.

11.1. Anwendungsfall 1: Produktion von Möbel

Anhand der Implementierung des ersten Anwendungsfalles (siehe Kapitel 10.1) kann zunächst eine 3D-Simulation einer Produktion gestartet sowie deren Spezifikationen definiert werden. Weiters kann über die TWAIN und das WfMS der Fortschritt der Produktion beobachtet und sogar abgebrochen werden. Über die Web API von Factory I/O sowie angesammelte produktionsrelevante Daten können Echtzeitdaten eingesehen und überwacht, aber auch Informationen aus historische Events bzw. Trends gewonnen werden.



Abbildung 11.3.: Schrank-Produktion der Fertigungslinie in Factory I/O (Screenshot)

11.1. Anwendungsfall 1: Produktion von Möbel

Abbildung 11.3 zeigt den Prozessweg zur Produktion von Schränken.

- ① kümmert sich um das Zuschneiden des Materials mit der dafür vorgesehenen Maschine. Weiters wird eine neue Palette vorbereitet, auf welcher dieselbe Referenznummer der ursprünglichen Palette auf Index 0 hinterlegt wird.
- ② beinhaltet Vorbohrarbeiten, welche, wie einige weitere Tasks, mit Hilfe einer Leuchte visualisiert werden, da in Factory I/O keine hierzu passenden Maschinen vorhanden sind.
- ③ simuliert die Herstellung der Regale.
- ④ zeigt die Fertigung der vorgesehenen Türart. Zur Visualisierung wird je nach Art eine unterschiedliche Farbe von der Leuchte geschaltet. Außerdem wird via MQTT ein Ereignis gesendet, anhand welchem schließlich eine Übersicht im Dashboard erstellt wird, welche die Häufigkeit der einzelnen Türarten aufzeigt.
- ⑤ baut ein Türschloss ein.
- ⑥ Bereitet eine neue Palette vor, auf welcher dieselbe Referenznummer der ursprünglichen Palette auf Index 0 hinterlegt wird. Außerdem wird das erforderliche Material (Spiegel) vorbereitet.
- ⑦ schneidet den Spiegel gemäß der Bestell- bzw. Simulationsdetails zu. Zur Visualisierung wird je nach Form eine unterschiedliche Farbe von der Leuchte geschaltet. Außerdem wird via MQTT ein Ereignis gesendet, anhand welchem schließlich eine Übersicht im Dashboard erstellt wird, welche die Häufigkeit der einzelnen Formen aufzeigt.
- ⑧ montiert den zugeschnittenen Spiegel an die hergestellte Tür.
- ⑨ legt Kleiderhaken sowie eine Schrankstange bei.

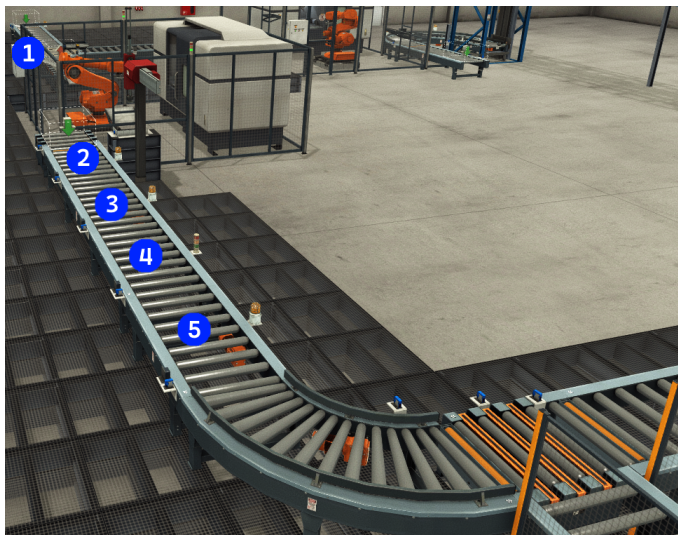


Abbildung 11.4.: Schreibtisch-Produktion der Fertigungslinie in Factory I/O (Screenshot)

11. Evaluierung

Abbildung 11.4 zeigt den Prozessweg zur Produktion von Schreibtischen.

- ① kümmert sich um das Zuschneiden des Materials mit der dafür vorgesehenen Maschine. Weiters wird eine neue Palette vorbereitet, auf welcher dieselbe Referenznummer der ursprünglichen Palette auf Index 0 hinterlegt wird.
- ② führt Vorbohrarbeiten aus.
- ③ simuliert die Herstellung der Schubladen.
- ④ bearbeitet die Tischplatte nach erforderlichlichem Stil (klassische Gravur oder Kabeldurchführung). Zur Visualisierung wird je nach Stil eine unterschiedliche Farbe von der Leuchte geschaltet. Außerdem wird via MQTT ein Ereignis gesendet, anhand welchen schließlich eine Übersicht im Dashboard erstellt wird, welche die Häufigkeit der einzelnen Stile aufzeigt.
- ⑤ erstellt die Tischbeine.

11.2. Anwendungsfall 2: Verpackung und Versand

Während bei Anwendungsfall 1 für sämtliche Entscheidungen die benötigten Informationen aus den Bestell- bzw. Simulationsdetails entnommen werden, werden beim Anwendungsfall 2 (siehe Kapitel 10.2) noch zusätzliche Werte von Sensoren sowie sonstigen Geräten miteinbezogen. So kann noch vor der Simulation eine Skala definiert werden, welche festlegt, wie einzelne Produkte kategorisiert und folglich verpackt und versendet werden.

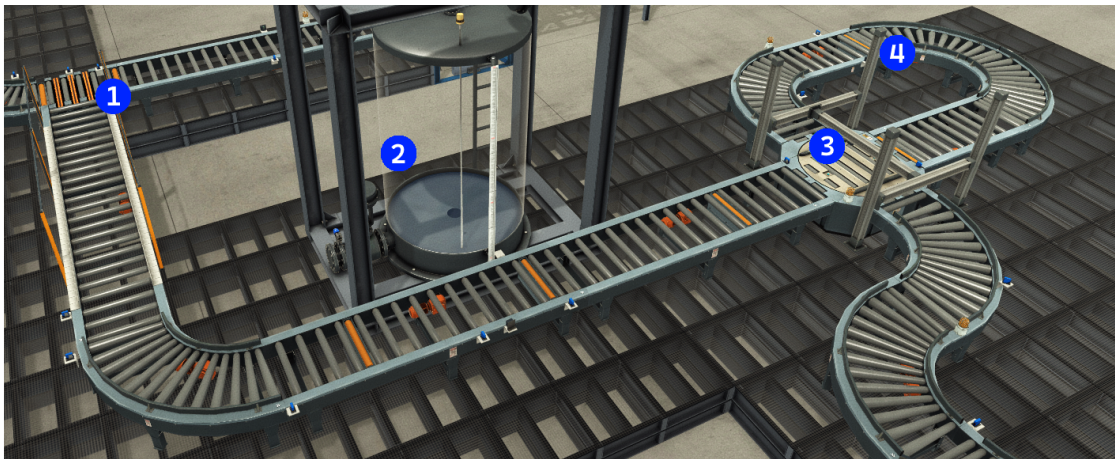


Abbildung 11.5.: Treffpunkt beider Produktionen der Fertigungslinie in Factory I/O (Screenshot)

11.2. Anwendungsfall 2: Verpackung und Versand

Abbildung 11.5 stellt dar, wie die Produktion nach dem erfolgreichen Durchqueren der Prozesspfade fortgesetzt wird.

- ① zeigt, wie sich beide Produktarten wieder auf demselben Fließband treffen.
- ② kümmert sich um die Oberflächenbehandlung der Produkte. Dabei wird der Tank-Level-Stand via MQTT übermittelt, wodurch dessen Trend schließlich im Dashboard ersichtlich ist.
- ③ führt eine Qualitätskontrolle durch, welche in 80% der Fälle positiv ausfällt. Dabei wird das Ergebnis via MQTT übermittelt, wodurch eine Status-Historie im Dashboard erstellt wird.
- ④ legt fehlende Bestandteile bei, wenn die Qualitätskontrolle negativ ausgefallen ist. Schließlich erfolgt eine erneute Qualitätskontrolle.

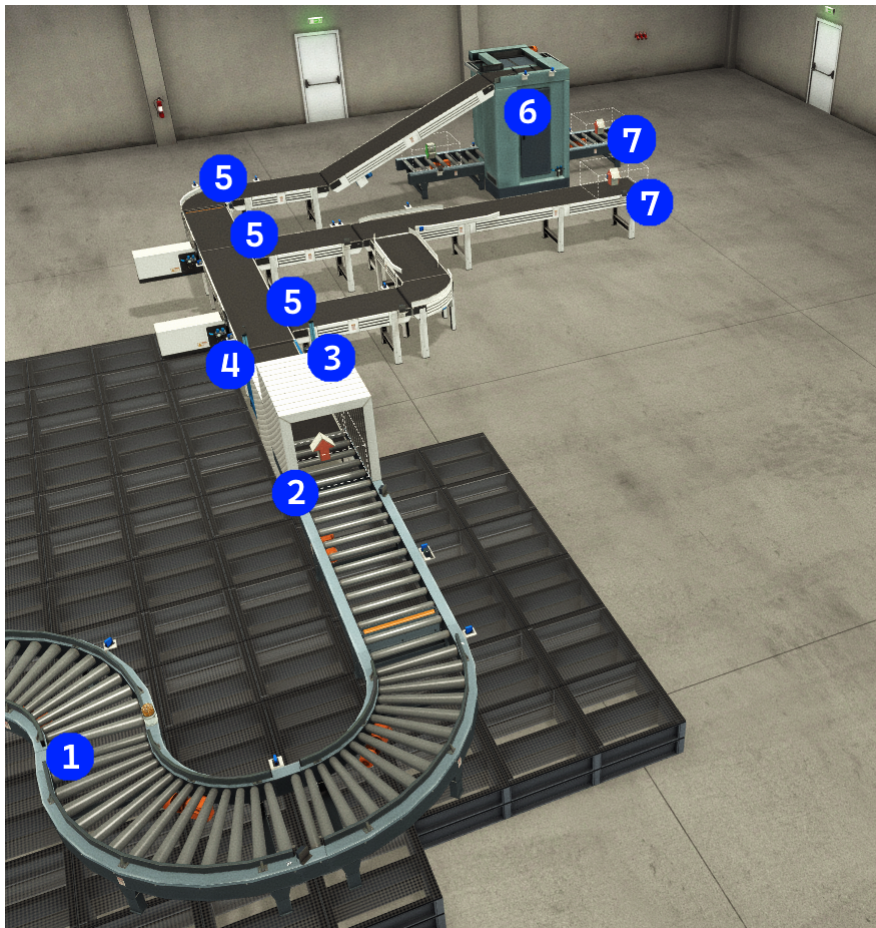


Abbildung 11.6.: Verpackung und Versand in Factory I/O (Screenshot)

11. Evaluierung

Abbildung 11.6 zeigt den abschließenden Verpackungs- und Versandprozess, welcher umgehend nach Anwendungsfall 1: Produktion von Möbel erfolgt.

- ① legt, wenn erforderlich, Zusatzausrüstung bei. Andernfalls wird dieser Task übersprungen (Leuchte wird nicht geschaltet).
- ② verpackt das Material im vorgesehenen Verpackungsmaterial. Weiters wird die Referenznummer der ursprünglichen Palette auf Index 0 des RFID-Transponders des Paketes hinterlegt. Das angewendete Verpackungsmaterial wird schließlich über MQTT gesendet, damit deren Häufigkeit in Form eines Diagrammes abgebildet werden kann.
- ③ misst die Höhe des Paketes. Dabei wird das Ergebnis via MQTT übermittelt, wodurch der Durchschnitt im Dashboard berechnet werden kann.
- ④ misst das Gewicht des Paketes. Dabei wird das Ergebnis via MQTT übermittelt, wodurch der Durchschnitt im Dashboard berechnet werden kann.
- ⑤ erstellt den erforderlichen Lieferschein basierend auf den Paketmaßen sowie Simulationsdetails.
- ⑥ stapelt zwei große Pakete auf einer Palette.
- ⑦ versendet das Produkt.

11.3. Ausführung einer prozessbasierten Simulation

Die auszuführende Simulation basiert auf einen zweiteiligen Prozess, welcher die Produktion und den Versand von zwei unterschiedlichen Produkten ermöglicht. Zur Ausführung von 3D-Simulation wird die virtuelle Fertigungslinie zunächst mittels der Software Factory I/O virtuell abgebildet. In jener können Sensoren, RFID-Lesegeräte sowie zahlreiche Produktionsgeräte platziert werden. Die daraus resultierende Fertigungslinie wird schließlich mit Hilfe eines PLC-Programmes automatisiert. Hierzu werden die PLC-Sprachen Funktionsplan (FUP) sowie strukturierter Text (ST) verwendet. Das kompilierte PLC-Programm kann anschließend via der *OpenPLC Runtime* als *Slave Device* zur Verfügung gestellt werden und folglich über den *Modbus TCP/IP Server* Treiber von Factory I/O verbunden werden [107]. Die von Factory I/O bereitgestellte Web API sowie SDK sorgen für hilfreiche Schnittstellen, anhand welcher die Steuerung anhand des WfMS schließlich realisiert werden kann [97]. Jene Steuerung erfolgt mit Hilfe des WfMS CPEE. Weiters sollen Echtzeitdaten in Form eines Dashboards bereitgestellt sowie historische Daten der Simulationen für zukünftige Analysen angesammelt werden. Hierzu werden Open Source Programme angewendet, wie InfluxDB als Echtzeitdatenbank sowie Grafana zur Darstellung jener Daten. Der hervorgehende digitale Zwilling soll schließlich von einem WfMS gesteuert werden können. Auf diese Weise kann eine individuelle Produktion durch das WfMS eingeleitet werden. Dadurch variiert der Fertigungsprozess aufgrund der spezifischen Fertigungsangaben.

Der Fertigungsprozess der Implementierung kurz zusammengefasst, beginnt bei beiden Produkten auf ein und demselben Fließband. Auf jenem wird das benötigte Material geladen. Im nächsten Schritt wird das Material zur passenden Produktionsmaschine

11.3. Ausführung einer prozessbasierten Simulation

gefördert, wodurch der Prozess zweigeteilt wird. Schließlich folgen weitere vom WfMS geregelte Produktionsschritte. Nachdem die Produktion abgeschlossen wurde, vereinen sich die beiden Prozesspfade erneut. Es folgen abschließende Lackierungs-, Inspektions-, Verpackungs- und Logistikarbeiten.

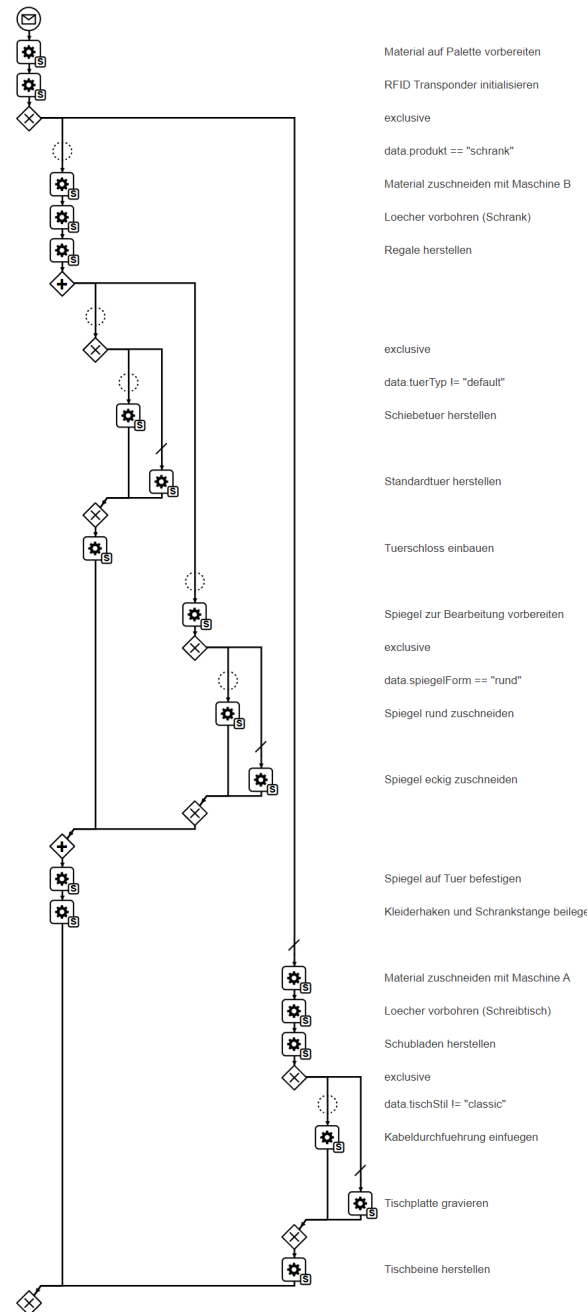


Abbildung 11.7.: Anwendungsfälle in CPEE (Screenshot Teil 1/2)

11. Evaluierung

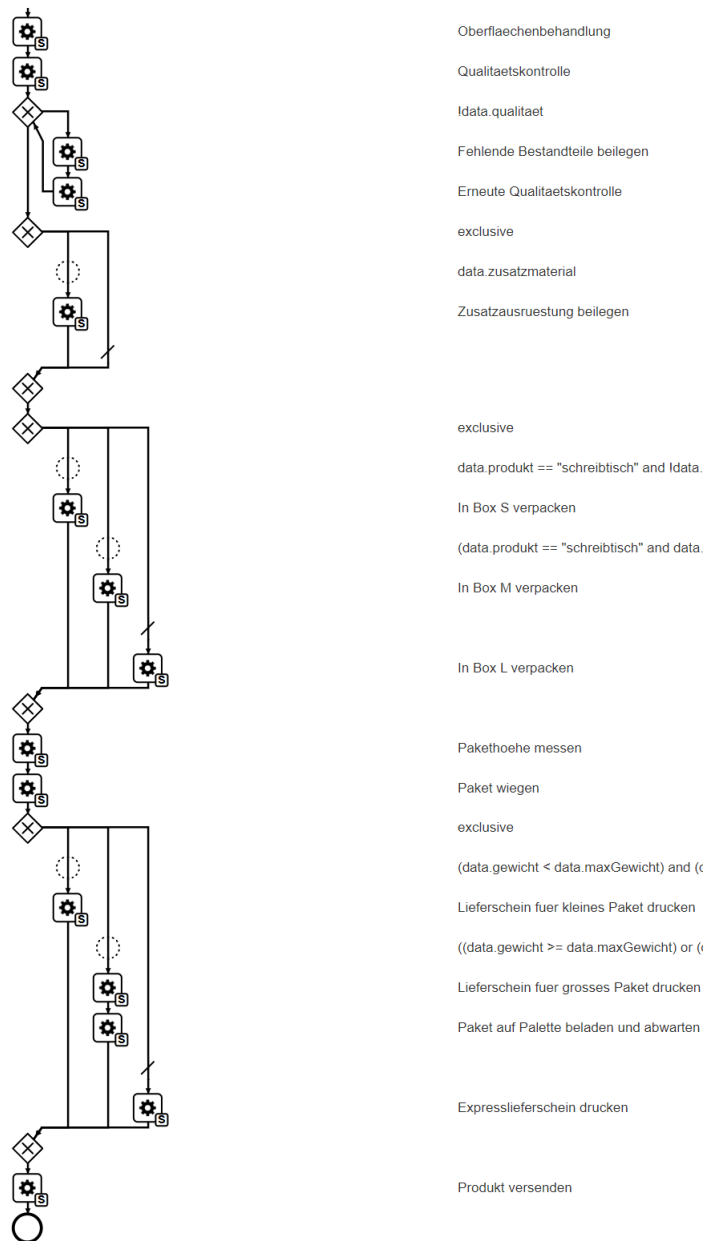


Abbildung 11.8.: Anwendungsfälle in CPEE (Screenshot Teil 2/2)

Abbildungen 11.7 und 11.8 zeigen das in CPEE abgebildete Prozessmodell, welches Anwendungsfall 1: Produktion von Möbel und Anwendungsfall 2: Verpackung und Versand hintereinander ausführt. Die einzelnen Tasks senden Requests an die Simulations-API und leuchten solange rot auf, bis diese erfolgreich abgeschlossen wurden. Anschließend werden die folgenden Tasks eingeleitet.

12. Conclusio

Das Ziel dieser Arbeit bestand darin, herauszufinden, in welchen Anwendungsgebieten WfMS und digitale Zwillinge gemeinsam zum Einsatz kommen. Weiters sollte ein digitaler Zwilling implementiert und ein Ansatz entwickelt werden, der es erlaubt, einen digitalen Zwilling anhand eines WfMS zu kontrollieren. Hierzu sollten zwei Anwendungsfälle in einer Fertigungsanlage gewählt und abgebildet werden, welche schließlich in Form einer Simulation durchgeführt werden können.

Im Rahmen dieser Arbeit konnte ein funktionsfähiges Konzept entwickelt, erfolgreich umgesetzt und anhand greifbarer Anwendungsfälle validiert werden. Im folgenden Abschnitt werden die Ergebnisse der Forschungsfragen behandelt.

F1 Wie werden digitale Zwillinge in Kombination mit WfMS genutzt?

Anhand der Literaturrecherche konnten einige typische Anwendungsbereiche digitaler Zwillinge herausarbeitet werden. Demnach werden digitale Zwillinge beispielsweise in Smart Cities, Smart Factories oder dem Gesundheitswesen eingesetzt. Im Rahmen dieser Abschlussarbeit wurde der Fokus auf Anwendungen in der Produktion gelegt. Aus der Literatur geht hervor, dass digitale Zwillinge in Kombination mit WfMS für verschiedene Zwecke genutzt werden. Rabah et al. [10] bezeichnen WfMS sogar als Rückgrat sowie zentrale Komponente digitaler Zwillinge, da anhand dieser Workflows von Nutzern definiert und modelliert, Geschäftsprozess-Fristen festgelegt, Szenarien simuliert, Grenzen vorausgesehen und KPIs überwacht werden können. Weiters können digitale Funktionen und Workflows automatisiert und reibungslos ausgeführt werden [10]. Bewährte WfMS Tools geben Nutzern die Möglichkeit, digitale Zwillinge einer Organisation zu erstellen, welche unter anderem der Analyse, Maßnahmenergreifung und Entscheidungsunterstützung dienen [7, 6]. Weiters ermöglichen prozessorientierte digitale Zwillingcockpits die Steuerung digitaler Zwillinge und vereinfachen die Modellierung, den Einsatz sowie die Überwachung solcher [92].

F2 Wie kann ein digitaler Zwilling anhand eines WfMS gesteuert werden?

WfMS können als zentrale Komponente zur Steuerung digitaler Zwillinge dienen. Um den digitalen Zwilling zu steuern, sendet das WfMS Befehle, um das System zu kontrollieren und zu überwachen.

WfMS ermöglichen die Definition und Modellierung von Workflows, anhand welcher Szenarien simuliert und automatisiert werden können. Dabei folgt das WfMS einem klar definierten Prozessablauf und trifft sämtliche Entscheidungen auf der Grundlage festgelegter Regeln. Dabei überwachen sie kontinuierlich wichtige Leistungsindikatoren KPIs. Im Rahmen der Implementierung wird die Steuerung des digitalen Zwillings anhand des WfMS mit Hilfe einer Simulations-API sowie der API und SDK von Factory I/O ermöglicht. Dabei werden einzelne Tasks des Workflows via Endpoints eingeleitet, welche wiederum über die API von Factory I/O Maschinen oder Geräte der Fertigungsline involvieren. Sobald ein Gerät oder eine Maschine deren Aufgabe erfüllt hat, wird mittels eines RFID-Lesers das Objekt identifiziert und erneut via einem HTTP PUT Request das WfMS informiert, dass der Task abgeschlossen wurde. In Folge dessen wird im WfMS der Task beendet und der nächste eingeleitet.

F3 Wie wird ein möglicher Ansatz zur Implementierung eines Prozesszwillings gestaltet?

Zur Realisierung einer solchen Implementierung wurde eine Microservice-Architektur angewendet. Hierzu wurden mit Hilfe von Docker separate Container erstellt. Neben jenen wird außerdem noch das 3D-Simulationsprogramm Factory I/O sowie deren API und SDK lokal ausgeführt. Zunächst wurden zwei Prozesse einer Fertigungsanlage definiert, in BPMN-Notation modelliert und im Simulationsprogramm abgebildet. Dementsprechend wurden dabei sämtliche notwendigen Sensoren angebracht, deren Werte schließlich über den digitalen Zwilling überwacht werden können. Zur Automatisierung der Fertigungsline wurde OpenPLC verwendet. Zudem wird anhand des SDK auf Werteänderung entsprechend reagiert, indem beispielsweise Informationen an das WfMS gesendet werden. Weiters werden Tasks über das WfMS sowie der API von Factory I/O eingeleitet und relevante Ereignisse via MQTT weiter an einen Kafka Broker bis hin zur Echtzeitdatenbank InfluxDB weitergeleitet. Jene dient schließlich als Datenquelle für Grafana, einem Tool zur Erstellung von Dashboards. Zur intuitiven Verwendung des digitalen Zwillings sowie zur Erstellung, Verwaltung und Ausführung von Simulationen wurde eine Web-Anwendung entwickelt.

F4 Welche Anwendungsfälle gibt es in Produktion und Logistik?

In dieser Arbeit wurden zwei von vielen potenziellen Anwendungsfällen eines WfMS-gesteuerten digitalen Zwillings vorgestellt und in Form eines Prototypen einer virtuellen Fertigungsline erfolgreich implementiert.

Anwendungsfall 1 *Produktion von Möbel* umfasst einen Produktionsprozess von zwei verschiedenen Produkten. Dabei wird der Produktionsablauf anhand der definierten Bestelldetails und dem WfMS entsprechend ausgeführt. Der zweite Anwendungsfall, *Verpackung und Versand*, basiert zusätzlich auf Spezifikationen hinsichtlich Verpackungsmaterial und Paketgrößen.

12.1. Ausblick

Ein vielversprechender nächster Schritt wäre die Aufzeichnung von Logs, die während der Fertigung generiert werden. Mithilfe von Process Mining könnte man nicht-triviale prozessbezogene Informationen aus diesen Ereignisdaten extrahieren [88]. Dies würde eine detailliertere Analyse der Produktionsabläufe ermöglichen und potenzielle neue Erkenntnisse über Prozessoptimierung und Effizienzsteigerung liefern. Ein weiterer Aspekt, der noch weiterentwickelt werden könnte, betrifft die Reaktion auf Fehler und unvorhergesehene Ereignisse. Beispielsweise könnte die Implementierung von Mechanismen in Betracht gezogen werden, um auf Kollisionen zwischen Paletten und Paketen oder auf Blockaden in der Fertigungslinie zu reagieren. Hierzu müsste definiert werden, wie solche Ereignisse erkannt und behandelt werden sollen. Dabei müsste man entscheiden, ob die gesamte Fertigungslinie abgeschaltet werden sollte oder nur einen Teil davon, um den Produktionsfluss aufrechterhalten zu können. Ein Aspekt, der außerhalb des Rahmens dieser Arbeit liegt, betrifft die Kommunikation zwischen dem digitalen Zwilling und seinem realen Gegenstück. Es wurde zwar die Kommunikation zwischen den virtuellen Objekten der Fertigungslinie und dem digitalen Zwilling simuliert, jedoch wäre es wünschenswert, dass ein reales Gegenstück existiert und über die vorgestellten Protokolle mit dem digitalen Zwilling kommuniziert.

Abkürzungsverzeichnis

- AAS** Verwaltungsschale, engl.: Administration Shell. 13
- AMQP** Advanced Message Queuing Protocol. 25
- API** Application Programming Interface. 25, 31, 32, 41, 45, 46, 48–54, 58–60, 69, 71, 74, 76, 77, 87, 90, 94, 96, 98
- AS** Ablaufsprache. 57
- AWL** Anweisungsliste. 57
- AWS** Amazon Web Services. 25
- BPM** Business Process Management. 27, 31, 36
- BPMN** Business Process Model and Notation. 27, 29, 31, 32, 36, 38, 41, 42, 44, 46, 59, 80, 83, 98
- BPMS** Business Process Management System. 29
- CPEE** Cloud Process Execution Engine. 4, 31, 32, 48, 50, 54, 59, 64, 70, 94, 96
- CPPS** Cyber-Physical-Production-System. 15
- CPS** Cyber-Physical-System. 14, 38
- DDMs** Datengesteuerte Modelle. 12
- DT4S** Digital Twin for Services. 3
- DTIS** Digitales Zwillingsinformationssystem. 38
- DTO** Digitaler Zwilling einer Organisation. 3, 20, 37
- FUP** Funktionsplan. 57, 94
- HTTP** Hypertext Transfer Protocol. 25, 30, 32, 41, 52, 53, 75, 98
- IIoT** Industrial Internet of Things. 15, 56
- IKT** Informations- und Kommunikationstechnologien. 14

Abkürzungsverzeichnis

- IoT** Internet der Dinge, engl.: Internet of Things. 1, 4, 7, 10–15, 17, 21–23, 25, 26, 41, 46, 49–51, 55–57, 60, 61, 79, 80, 83
- JSON** JavaScript Object Notation. 46, 64, 80
- KI** Künstliche Intelligenz, engl.: Artificial Intelligence, AI. 1, 7, 8, 10, 12–14, 21, 22, 26
- KOP** Kontaktplan. 57
- KPI** Leistungskennzahlen, engl.: Key Performance Indicator. 39, 97, 98
- M2M2P** Maschine-zu-Maschine-zu-Mensch. 21
- MEMS** Micro-Electro-Mechanical-System. 11
- ML** Maschinelles Lernen, engl.: Machine Learning. 10, 22
- MQTT** Message Queuing Telemetry Transport. 25, 49–52, 55, 56, 61, 71, 73, 80, 83, 87, 89, 91–94, 98
- OPA** Open Process Automation. 39
- OPC-UA** Open Platform Communications Unified Architecture. 56
- PaaS** Plattform-as-a-Service. 25
- PADTC** Process-aware digital twin cockpit. 38
- PAIS** Process-Aware Information Systems. 27, 28, 37
- PLC** Programmable Logic Controller. 3, 4, 24, 26, 41, 44, 48, 51, 52, 57, 77, 87, 89, 94
- PLM** Product Lifecycle Management. 10
- PoT** Prozess der Dinge, engl.: Process of Things. 4
- RAMI4.0** Referenzarchitekturmodell Industrie 4.0. 55
- REST** Representational State Transfer. 31–33, 53, 59, 87
- RFID** Radio-Frequency Identification. 42, 44, 53, 54, 56, 59, 72, 79, 80, 83, 87, 89, 90, 94, 98
- SDK** Software Development Kit. 24, 41, 48, 50–53, 56, 58, 59, 66, 71, 72, 75–77, 87, 89, 90, 94, 98
- SMS** Intelligentes Fertigungssystem, Smart Manufacturing System. 14
- SPS** Speicherprogrammierbare Steuerung. 3, 57

ST Strukturierter Text. 57, 94

TWAIN Digital Twin Simulation Application. 43–45, 48, 49, 51–54, 60–65, 70, 71, 77, 80, 83, 87, 90

UI User Interface. 60, 70

VM Virtuelle Maschine. 66

WfMS Workflow-Management-System. iii, v, 3–5, 27–29, 31, 32, 34–37, 39, 41–49, 51–54, 59, 64, 71, 74–77, 79, 80, 87, 90, 94, 95, 97–99

XML Extensible Markup Language. 54

Literaturverzeichnis

- [1] **Kuhn, T.** *Digitaler Zwilling*, volume 40. Informatik Spektrum, 2017. doi: 10.1007/s00287-017-1061-2. URL <https://doi.org/10.1007/s00287-017-1061-2>.
- [2] **Sauer, O.** *The Digital Twin – An Essential Key Technology for Industrie 4.0*, volume 115. ERCIM News, 2018. URL <https://ercim-news.ercim.eu/images/stories/EN115/EN115-web.pdf>.
- [3] **Qi, Q. and Tao, F.** *Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison*, volume 6. 2018. doi: 10.1109/ACCESS.2018.2793265. URL <https://doi.org/10.1109/ACCESS.2018.2793265>.
- [4] **Guerra-Zubiaga, D., Kuts, V., Mahmood, K., Bondar, A., Nasajpour-Esfahani, N., Otto, T.** *An approach to develop a digital twin for industry 4.0 systems: manufacturing automation case studies*, volume 34. 07 2021. doi: 10.1080/0951192X.2021.1946857. URL <https://doi.org/10.1080/0951192X.2021.1946857>.
- [5] **Dumas, M., La Rosa, M., Mendling, J., Reijers, H.** *Introduction to Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. ISBN 978-3-662-56509-4. doi: 10.1007/978-3-662-56509-4_1. URL https://doi.org/10.1007/978-3-662-56509-4_1.
- [6] **Signavio.** *Visualizing your Digital Process Twin*. <https://www.signavio.com/downloads/short-reads/visualizing-your-digital-process-twin/>, . Zuletzt abgerufen am 05.12.2022.
- [7] **BOC Group.** *ADONIS Process Simulation*. <https://knowledge.boc-group.com/de/module/adonis-process-simulation/#>, 2019. Zuletzt abgerufen am 05.12.2022.
- [8] **Siemens.** *Optimize production planning to ensure smooth operation*. <https://new.siemens.com/global/en/markets/automotive-manufacturing/digital-twin-production.html>, . Zuletzt abgerufen am 07.12.2022.
- [9] **Microsoft.** *The Process Digital Twin: A step towards operational excellence*. <https://info.microsoft.com/rs/157-GQE-382/images/Digital%20Twin%20Vision.pdf>, 2017. Zuletzt abgerufen am 05.12.2022.
- [10] **Rabah, S., Zacharewicz, G., Chapurlat, V.** *Digital Twin for Services (DT4S): Conceptual Strategy*. In *10th IFAC Conference on Manufacturing Modelling, Management and Control (MIM) 2022*, pages 3256–3261, Nantes, France,

- June 2022. IMT Atlantique, Nantes. doi: 10.1016/j.ifacol.2022.10.136. URL <https://hal.archives-ouvertes.fr/hal-03834868>. Published in : IFAC-PapersOnLine 55(10):3256–3261. doi :10.1016/j.ifacol.2022.10.136.
- [11] **Rinderle-Ma, S., Mangler, J.** Process automation and process mining in manufacturing. In *Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings*, pages 3—14, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-85468-3. doi: 10.1007/978-3-030-85469-0_1. URL https://doi.org/10.1007/978-3-030-85469-0_1.
 - [12] **Maamar, Z., Sellami, M., Faci, N., Ugljanin, E., Sheng, Q.,.** *Storytelling Integration of the Internet of Things into Business Processes*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-98651-7. doi: 10.1007/978-3-319-98651-7_8. URL https://doi.org/10.1007/978-3-319-98651-7_8.
 - [13] **Kamath, V., Morgan, J., Ali, M.** *Industrial IoT and Digital Twins for a Smart Factory : An open source toolkit for application design and benchmarking*. 2020. doi: 10.1109/GIOTS49054.2020.9119497. URL <https://doi.org/10.1109/GIOTS49054.2020.9119497>.
 - [14] **Alves, T., Buratto, M., Souza, F., Rodrigues, T.** *OpenPLC: An open source alternative to automation*. 10 2014. doi: 10.1109/GHTC.2014.6970342. URL <https://doi.org/10.1109/GHTC.2014.6970342>.
 - [15] **Gartner.** *Gartners Top 10 Technology Trends 2017*. <https://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017>, 2016. Zuletzt abgerufen am 16.08.2022.
 - [16] **Gartner.** *Gartner Identifies the Top 10 Strategic Technology Trends for 2018*. <https://www.gartner.com/en/newsroom/press-releases/2017-10-04-gartner-identifies-the-top-10-strategic-technology-trends-for-2018>, 2017. Zuletzt abgerufen am 16.08.2022.
 - [17] **McKinsey.** *Digital twins: Form one twin to the enterprise metaverse*. <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/digital-twins-from-one-twin-to-the-enterprise-metaverse>, 2022. Zuletzt abgerufen am 05.12.2022.
 - [18] **Köhler, H.** *Spiegelbild mit Potenzial*, volume 3. T-Systems, 2018. URL <https://www.t-systems.com/de/de/newsroom/best-practice/03-2018-digitaler-zwilling>.
 - [19] **Lu, Y., Liu, C., Wang, K., Huang, H., Xu, X.** *Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues*. *Robotics and Computer-Integrated Manufacturing*, 61:101837, 2020. ISSN 0736-5845. doi: <https://doi.org/10.1016/j.rcim.2019.101837>. URL <https://www.sciencedirect.com/science/article/pii/S0736584519302480>.

- [20] **Ding, K., Chan, F., Zhang, X., Zhou, G., Zhang, F.** *Defining a Digital Twin-based Cyber-Physical Production System for Autonomous Manufacturing in Smart Shop Floors.* *International Journal of Production Research*, 1:1–23, 01 2019. doi: 10.1080/00207543.2019.1566661. URL <https://doi.org/10.1080/00207543.2019.1566661>.
- [21] **Zheng, Y., Yang, S., Cheng, H.** *An application framework of digital twin and its case study.* *Journal of Ambient Intelligence and Humanized Computing*, 10, 03 2019. doi: 10.1007/s12652-018-0911-3. URL <https://doi.org/10.1007/s12652-018-0911-3>.
- [22] **Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., Sui, F.** *Digital twin-driven product design, manufacturing and service with big data.* *The International Journal of Advanced Manufacturing Technology*, 94, 02 2018. doi: 10.1007/s00170-017-0233-1. URL <https://doi.org/10.1007/s00170-017-0233-1>.
- [23] **Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.** *Digital Twin in manufacturing: A categorical literature review and classification.* *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. ISSN 2405-8963. doi: 10.1016/j.ifacol.2018.08.474. URL <https://www.sciencedirect.com/science/article/pii/S2405896318316021>. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.
- [24] **Negri, E., Fumagalli, L., Macchi, M.** *A Review of the Roles of Digital Twin in CPS-based Production Systems.* *Procedia Manufacturing*, 11:939–948, 2017. ISSN 2351-9789. doi: 10.1016/j.promfg.2017.07.198. URL <https://www.sciencedirect.com/science/article/pii/S2351978917304067>. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.
- [25] **Boschert, S., Rosen, R.** *Digital Twin—The Simulation Aspect*, pages 59–74. 06 2016. ISBN 978-3-319-32154-7. doi: 10.1007/978-3-319-32156-1_5. URL https://doi.org/10.1007/978-3-319-32156-1_5.
- [26] **Gabor, T., Belzner, L., Kiermeier, M., Beck, M., Neitz, A.** *A Simulation-Based Architecture for Smart Cyber-Physical Systems.* In *2016 IEEE International Conference on Autonomic Computing (ICAC)*, pages 374–379, 2016. doi: 10.1109/ICAC.2016.29. URL <https://doi.org/10.1109/ICAC.2016.29>.
- [27] **Rosen, R., von Wichert, G., Lo, G., Bettenhausen, K.** *About The Importance of Autonomy and Digital Twins for the Future of Manufacturing.* volume 48, pages 567–572, 2015. doi: <https://doi.org/10.1016/j.ifacol.2015.06.141>. URL <https://www.sciencedirect.com/science/article/pii/S2405896315003808>. 15th IFAC Symposium on Information Control Problems in Manufacturing.
- [28] **Grieves, M.** *Digital Twin: Manufacturing Excellence through Virtual Factory Replication.*

- White paper*, 1:1–7, 2014. URL https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication.
- [29] Gockel, B., Tudor, A., Brandyberry, M., Ravi, C., Tuegel, E. *Challenges with Structural Life Forecasting using Realistic Mission Profiles*. 2012. doi: 10.2514/6.2012-1813. URL <https://doi.org/10.2514/6.2012-1813>. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference.
- [30] Shafto, M., Conroy, M., Doyle, R., Glaessgen, E. Kemp, C., LeMoigne, J., Wang, L. *Modeling, Simulation, Information Technology and Processing Roadmap*. 05 2010. URL https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf.
- [31] Bjorkberg, T. *Digital Twin, a Mirrored World*. <https://www.zte.com.cn/global/about/magazine/zte-technologies/2022/4-en/expert-views/3.html>, 2022. Zuletzt abgerufen am 20.12.2022.
- [32] IBM. *What is a digital twin?* <https://www.ibm.com/topics/what-is-a-digital-twin>. Zuletzt abgerufen am 05.12.2022.
- [33] Boschert, S., Rosen, R. *Digital Twin: a Second Life for Engineering Models*. 2018. URL <https://ercim-news.ercim.eu/images/stories/EN115/EN115-web.pdf>.
- [34] Waffenschmidt, S. *Digitaler Zwilling*, volume 3. T-Systems, 2018. URL <https://www.t-systems.com/de/de/newsroom/best-practice/03-2018-digitaler-zwilling>.
- [35] Krack, Markus and Burri, Simona and Fischer, Manuel. *Der Digital Twin. Das unbekannte Wesen. Technische Rundschau*, 113:32–34, 2021. doi: 10.26041/fhnw-3867. URL <https://doi.org/10.26041/fhnw-3867>.
- [36] Industrie2025. *Arbeitsgruppe Digitaler Zwilling*. <https://www.industrie2025.ch/wissen-industrie-40/arbeitsgruppen/digitaler-zwilling>, 2021. Zuletzt abgerufen am 20.12.2022.
- [37] Al-Ali, A., Gupta, R., Zaman Batool, T., Landolsi, T., Aloul, F., Al Nabulsi, A. *Digital Twin Conceptual Model within the Context of Internet of Things. Future Internet*, 12(10), 2020. ISSN 1999-5903. doi: 10.3390/fi12100163. URL <https://www.mdpi.com/1999-5903/12/10/163>.
- [38] Manca, L., Grugni, R., Mirzazadeh, R. *Digital Twin*. https://www.eng.it/resources/whitepaper/doc/digital-twin/digital-twin_whitepaper_en.pdf, 2020. Zuletzt abgerufen am 20.12.2022.

- [39] **Nguyen, H., Trestian, R., To, D., Tatipamula, M.** *Digital Twin for 5G and Beyond*, volume 59. 2021. doi: 10.1109/MCOM.001.2000343. URL <https://doi.org/10.1109/MCOM.001.2000343>.
- [40] **Rathore, M., Shah, S., Shukla, D., Bentafat, E., Bakiras, S.** *The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities*. *IEEE Access*, 9:32030–32052, 2021. doi: 10.1109/ACCESS.2021.3060863. URL <https://doi.org/10.1109/ACCESS.2021.3060863>.
- [41] **Plattform Industrie 4.0 - Bundesministerium für Wirtschaft und Klimaschutz.** *Der digitale Zwilling in der Industrie 4.0*. https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/presentation_der-digitale-Zwilling.html, 2020. Zuletzt abgerufen am 07.12.2022.
- [42] **Siemens.** *Digitale Zwillinge*. <https://new.siemens.com/de/de/unternehmen/stories/forschung-technologien/digitaler-zwilling/digital-twin.html>, 2021. Zuletzt abgerufen am 07.12.2022.
- [43] **VDE.** *Der Digitale Zwilling ist Wegbereiter für alle Industrie-4.0-Netze*. <https://www.vde.com/topics-de/industry/aktuelles/digitaler-zwilling-wegbereiter-fuer-industrie-40>, 2018. Zuletzt abgerufen am 07.12.2022.
- [44] **Kunz, D.** *Ohne digitale Zwillinge keine smarte Fabrik*. <https://iot.telekom.com/de/blog/ohne-digitale-zwillinge-keine-smarte-fabrik>, 2021. Zuletzt abgerufen am 07.12.2022.
- [45] **Pires, F., Cachada, A., Barbosa, J., Moreira, A., Leitão, P.** *Digital Twin in Industry 4.0: Technologies, Applications and Challenges*. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 721–726, 2019. doi: 10.1109/INDIN41052.2019.8972134. URL <https://doi.org/10.1109/INDIN41052.2019.8972134>.
- [46] **Jiewu, L., Dewen, W., Weiming, S., Xinyu, L., Qiang, L., Xin, C.** *Digital twins-based smart manufacturing system design in Industry 4.0: A review*. *Journal of Manufacturing Systems*, 60:119–137, 2021. ISSN 0278-6125. doi: <https://doi.org/10.1016/j.jmsy.2021.05.011>. URL <https://www.sciencedirect.com/science/article/pii/S0278612521001151>.
- [47] **Lattanzi, L., Raffaeli, R., Peruzzini, M., Pellicciari, M.** *Digital twin for smart manufacturing: a review of concepts towards a practical industrial implementation*. *International Journal of Computer Integrated Manufacturing*, 34(6):567–597, 2021. doi: 10.1080/0951192X.2021.1911003. URL <https://doi.org/10.1080/0951192X.2021.1911003>.

- [48] **Microsoft**. *IoT Signals Manufacturing Spotlight*. Microsoft, 2022. URL <https://info.microsoft.com/ww-landing-IoT-signals-manufacturing-spotlight.html?lcid=en-us>.
- [49] **Damjanovic-Behrendt, V., Behrendt, W.** *An open source approach to the design and implementation of Digital Twins for Smart Manufacturing*. *International Journal of Computer Integrated Manufacturing*, 32(4-5):366–384, 2019. doi: 10.1080/0951192X.2019.1599436. URL <https://doi.org/10.1080/0951192X.2019.1599436>.
- [50] **Germanedge**. *Smart Factory als Teil der Produktion der Zukunft*. <https://www.germanedge.com/zukunft-der-produktion/smart-factory/>, 2020. Zuletzt abgerufen am 20.12.2022.
- [51] **Burke, R., Laaper, S., Hartigan, M., Sniderman, B.** *The smart factory*. Deloitte, 2017. URL <https://www2.deloitte.com/us/en/insights/focus/industry-4-0/smart-factory-connected-manufacturing.html>.
- [52] **TWI**. *Simulation vs Digital Twin (What is the difference between them?)*. <https://www.twi-global.com/technical-knowledge/faqs/simulation-vs-digital-twin#SimulationVsDigitalTwinComparison>. Zuletzt abgerufen am 07.12.2022.
- [53] **Herrlich, B.** *Smarte Simulation der Wirklichkeit: Digital Twins*. <https://www.t-systems.com/at/de/newsroom/blog-alpine/smart-simulation-der-wirklichkeit-digital-twins-359548>, 2020. Zuletzt abgerufen am 05.12.2022.
- [54] **Köhler, H.** *Die Fabrik und ihr Zwilling*, volume 3. T-Systems, 2018. URL <https://www.t-systems.com/de/de/newsroom/best-practice/03-2018-digitaler-zwilling>.
- [55] **Krieger, W.** *Produktionslogistik*. <https://wirtschaftslexikon.gabler.de/definition/produktionslogistik-43280>. Zuletzt abgerufen am 07.12.2022.
- [56] **Botín-Sanabria, D., Mihaita, A., Peimbert-García, R., Ramírez-Moreno, M., Ramírez-Mendoza, R., Lozoya-Santos, J.** *Digital Twin Technology Challenges and Applications: A Comprehensive Review*, volume 14. 2022. doi: 10.3390/rs14061335. URL <https://www.mdpi.com/2072-4292/14/6/1335>.
- [57] **Chaplin, J., Martinez-Arellano, G., Mazzoleni, A.** *Digital Twins and Intelligent Decision Making*, 2020. URL <https://nottingham-repository.worktribe.com/output/5312833>. ISBN not found on internet search and does not appear on the online version of the book.
- [58] **Gartner**. *Quick Answer: What Is a Digital Twin of an Organization?* <https://www.gartner.com/en/documents/4004172>, 2021. Zuletzt abgerufen am 20.12.2022.

- [59] **Parrott, A., Warshaw, L.** *Industry 4.0 and the digital twin*. <https://www2.deloitte.com/us/en/insights/focus/industry-4-0/digital-twin-technology-smart-factory.html>, 2017. Zuletzt abgerufen am 19.04.2023.
- [60] **Gya, R., Petit, J., Bacry, J., Bronson, B., Capone, A., Lange, U., Gertz, V., Crou  , N., Mezhr  hid, J., Berg, A., Naik, N., Vasilyeva, E., Buvat, J., Kanakadandi, S., Puttur, R., Pande, S.** *Digital Twins: Adding Intelligence to the real World*. https://prod.ucwe.capgemini.com/wp-content/uploads/2022/05/Capgemini-Research-Institute_DigitalTwins_Web.pdf, 2022. Zuletzt abgerufen am 03.01.2023.
- [61] **Balachandra, V., Prahallad, C.** *Digital Twin White Paper*. https://www.bosch-softwaretechnologies.com/en/services/enterprise-services/manufacturing-excellence/digital_twin/, 2020. Zuletzt abgerufen am 03.01.2023.
- [62] **NVIDIA.** *Digitale Zwillinge*. <https://www.nvidia.com/de-de/omniverse/solutions/digital-twins/>. Zuletzt abgerufen am 03.01.2023.
- [63] **B  c  ue, A., Maia, E., Feeken, L., Borchers, P., Pra  a, I.** *A New Concept of Digital Twin Supporting Optimization and Resilience of Factories of the Future*. *Applied Sciences*, 10(13), 2020. ISSN 2076-3417. doi: 10.3390/app10134482. URL <https://www.mdpi.com/2076-3417/10/13/4482>.
- [64] **Siemens.** *NX*. <https://www.plm.automation.siemens.com/global/de/products/nx/>, . Zuletzt abgerufen am 03.01.2023.
- [65] **Eclipse.** *Ditto*. <https://www.eclipse.org/ditto/>, . Zuletzt abgerufen am 03.01.2023.
- [66] **Eclipse.** *Eclipse Ditto Documentation*. <https://www.eclipse.org/ditto/intro-overview.html>, . Zuletzt abgerufen am 03.01.2023.
- [67] **Eclipse.** *Eclipse Ditto*. <https://projects.eclipse.org/projects/iot.ditto>, . Zuletzt abgerufen am 03.01.2023.
- [68] **AWS.** *AWS IoT*. <https://aws.amazon.com/de/iot/>, . Zuletzt abgerufen am 03.01.2023.
- [69] **AWS.** *AWS IoT TwinMaker*. <https://aws.amazon.com/de/iot-twinmaker/>, . Zuletzt abgerufen am 03.01.2023.
- [70] **Microsoft Azure.** *Azure Digital Twins*. <https://azure.microsoft.com/en-us/products/digital-twins/#overview>, . Zuletzt abgerufen am 03.01.2023.

- [71] **Microsoft Azure.** *What is Azure Digital Twins?*
<https://learn.microsoft.com/en-us/azure/digital-twins/overview>, 2022.
Zuletzt abgerufen am 03.01.2023.
- [72] **Microsoft Azure.** *Build in the cloud with an Azure free account.*
<https://azure.microsoft.com/en-us/free/>, . Zuletzt abgerufen am 03.01.2023.
- [73] **FlexSim Software Products.** *FlexSim.*
<https://www.flexsim.com/flexsim/>, . Zuletzt abgerufen am 03.01.2023.
- [74] **FlexSim Software Products.** *FlexSim + Digital Twin .*
<https://www.flexsim.com/digital-twin/>, . Zuletzt abgerufen am 03.01.2023.
- [75] **FlexSim Software Products.** *Students.*
<https://www.flexsim.com/students/>, . Zuletzt abgerufen am 03.01.2023.
- [76] **Factory I/O.** *About.* <https://docs.factoryio.com/#about>, . Zuletzt abgerufen am 16.08.2022.
- [77] **Breuer, H.** Überall digitale zwillinge. <https://new.siemens.com/de/de/unternehmen/stories/forschung-technologien/digitaler-zwilling/outlook2030.html>, 2020. Zuletzt abgerufen am 16.08.2022.
- [78] **OMG.** *Business Process Model and Notation (BPMN).* <https://www.omg.org/spec/BPMN/2.0/PDF>, 2011. Zuletzt abgerufen am 03.01.2023.
- [79] **AristaFlow.** *AristaFlow BPM Documentation.*
<https://docs.aristaflow.com/Latest/index.de/>, . Zuletzt abgerufen am 25.12.2022.
- [80] **AristaFlow.** *Registration for Research Education License.*
<https://www.aristaflow.com/request-re-lizenz.html>, . Zuletzt abgerufen am 25.12.2022.
- [81] **Signavio.** *SAP Signavio Process Manager.*
<https://documentation.signavio.com/suite/de/Content/process-manager.htm>, . Zuletzt abgerufen am 25.12.2022.
- [82] **ADONIS.** *Das ADONIS 15.0 Benutzerhandbuch.*
https://docs.boc-group.com/adonis/de/docs/15.0/user_manual/. Zuletzt abgerufen am 25.12.2022.
- [83] **Bonitasoft.** *Bonita Documentation.*
<https://documentation.bonitasoft.com/bonita/2022.2/>, 2022. Zuletzt abgerufen am 25.12.2022.
- [84] **CPEE.** *Cloud Process Execution Engine.* <https://cpee.org/index.php>. Zuletzt abgerufen am 16.08.2022.

- [85] **Van der Aalst, W.** *Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques*, pages 161–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. ISBN 978-3-540-45594-3. doi: 10.1007/3-540-45594-9_11. URL https://doi.org/10.1007/3-540-45594-9_11.
- [86] **Van der Aalst, W.** *Verification of Workflow nets*. pages 407–426, 01 1997. ISBN 3-540-63139-9. doi: 10.1007/3-540-63139-9_48. URL https://doi.org/10.1007/3-540-63139-9_48.
- [87] **Van der Aalst, W.** *Putting high-level Petri nets to work in industry*. *Computers in Industry*, 25(1):45–54, 1994. ISSN 0166-3615. doi: [https://doi.org/10.1016/0166-3615\(94\)90031-0](https://doi.org/10.1016/0166-3615(94)90031-0). URL <https://www.sciencedirect.com/science/article/pii/0166361594900310>.
- [88] **Sato, D., De Freitas, S., C., Barddal, J., P., Scalabrin, E.** *A Survey on Concept Drift in Process Mining*. *ACM Comput. Surv.*, 54(9), oct 2021. ISSN 0360-0300. doi: 10.1145/3472752. URL <https://doi.org/10.1145/3472752>.
- [89] **Hill, T.** *Die Leistung von Fertigungsprozessen überwachen und optimieren*. https://cdn.signavio.com/uploads/2021/12/Produzierendes-Gewerbe_Whitepaper_Signavio_2020.pdf, 2020. Zuletzt abgerufen am 05.12.2022.
- [90] **Park, G., Van Der Aalst, W.** *Realizing A Digital Twin of An Organization Using Action-oriented Process Mining*. In *2021 3rd International Conference on Process Mining (ICPM)*, pages 104–111, 2021. doi: 10.1109/ICPM53251.2021.9576846. URL <https://doi.org/10.1109/ICPM53251.2021.9576846>.
- [91] **Drave, I., Michael, J., Müller, E., Rumpe, B., Varga, S.** *Model-Driven Engineering of Process-Aware Information Systems*. *SN Computer Science*, 3(479), 2022. doi: 10.1007/s42979-022-01334-3. URL <https://doi.org/10.1007/s42979-022-01334-3>.
- [92] **Dalibor, M., Michael, J., Rumpe, B., Varga, S., Wortmann, A.** *Towards a Model-Driven Architecture for Interactive Digital Twin Cockpits*. In *Conceptual Modeling*, pages 377–387, Cham, 2020. Springer International Publishing. ISBN 978-3-030-62522-1. URL https://doi.org/10.1007/978-3-030-62522-1_28.
- [93] **Kirchhof, J. and Michael, J., Rumpe, B., Varga, S., Wortmann, A.** *Model-Driven Digital Twin Construction: Synthesizing the Integration of Cyber-Physical Systems with Their Information Systems*. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS '20*, page 90–101, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370196. doi: 10.1145/3365438.3410941. URL <https://doi.org/10.1145/3365438.3410941>.
- [94] **Bano, D., Michael, J., Rumpe, B., Varga, S., Weske, M.** *Process-aware digital twin cockpit synthesis from event logs*. *Journal of Computer Languages*, 70:

- 101121, 2022. ISSN 2590-1184. doi: 10.1016/j.cola.2022.101121. URL <https://www.sciencedirect.com/science/article/pii/S2590118422000259>.
- [95] **IONOS**. *Low Code: Definition, Besonderheiten und Einsatzgebiete*. <https://www.ionos.at/digitalguide/websites/web-entwicklung/was-ist-low-code/>, 2020. Zuletzt abgerufen am 03.01.2023.
- [96] **Qamsane, Y., Phillips, J., Savaglio, C., Warner, D., James, S., Barton, K.** *Open Process Automation- and Digital Twin-Based Performance Monitoring of a Process Manufacturing System*. *IEEE Access*, 10:60823–60835, 2022. doi: 10.1109/ACCESS.2022.3179982. URL <https://doi.org/10.1109/ACCESS.2022.3179982>.
- [97] **Factory I/O**. *Manual*. <https://docs.factoryio.com/manual>, . Zuletzt abgerufen am 16.08.2022.
- [98] **Muralidharan, S., Yoo, B., Ko, H.** *Designing a Semantic Digital Twin model for IoT*. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2, 2020. doi: 10.1109/ICCE46568.2020.9043088. URL <https://doi.org/10.1109/ICCE46568.2020.9043088>.
- [99] **Redhat**. *Was sind Microservices?* <https://www.redhat.com/de/topics/microservices/what-are-microservices>, 2023. Zuletzt abgerufen am 08.04.2023.
- [100] **Docker**. *What is a Container?* <https://www.docker.com/resources/what-container/>, 2023. Zuletzt abgerufen am 08.04.2023.
- [101] **AWS**. *Was ist MQTT?* <https://aws.amazon.com/de/what-is/mqtt/>, 2023. Zuletzt abgerufen am 06.04.2023.
- [102] **Bances, E., Schneider, U., Siegert, J., Bauernhansl, T.** *Exoskeletons Towards Industrie 4.0: Benefits and Challenges of the IoT Communication Architecture*. *Procedia Manufacturing*, 42:49–56, 2020. ISSN 2351-9789. doi: 10.1016/j.promfg.2020.02.087. URL <https://www.sciencedirect.com/science/article/pii/S2351978920306521>. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019).
- [103] **Plattform Industrie 4.0 - Bundesministerium für Wirtschaft und Klimaschutz**. *RAMI 4.0 – Ein Orientierungsrahmen für die Digitalisierung*. <https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/rami40-eine-einfuehrung.html>, 2018. Zuletzt abgerufen am 06.04.2023.
- [104] **Confluent**. *Apache Kafka Native MQTT at Scale with Confluent Cloud and Waterstream*. <https://www.confluent.io/blog/iot-streaming-use-cases-with-kafka-mqtt-confluent-and-waterstream/>, 2020. Zuletzt abgerufen am 08.04.2023.
- [105] **Apache Kafka**. *Apache Kafka*. <https://kafka.apache.org/>, 2023. Zuletzt abgerufen am 08.04.2023.

- [106] **Waehner, K.** *Apache Kafka + MQTT = End-to-End IoT Integration*. <https://www.kai-waehner.de/blog/2018/09/10/apache-kafka-mqtt-end-to-end-iot-integration-demo-scripts-github/>, 2018. Zuletzt abgerufen am 08.04.2023.
- [107] **OpenPLC.** *OpenPLC Overview*. <https://openplcproject.com/docs/openplc-overview/>, 2022. Zuletzt abgerufen am 16.08.2022.
- [108] **3S-Smart Software Solutions GmbH.** *Handbuch für SPS Programmierung mit CoDeSys 2.3*. Kempten, Deutschland, 2010.
- [109] **InfluxDB.** *Products*. <https://www.influxdata.com/products/>, 2023. Zuletzt abgerufen am 12.04.2023.
- [110] **InfluxDB.** *InfluxDB*. <https://www.influxdata.com/products/influxdb/>, 2023. Zuletzt abgerufen am 12.04.2023.
- [111] **InfluxDB.** *How InfluxDB Works with IoT Data*. <https://www.influxdata.com/blog/how-influxdb-iot-data/>, 2021. Zuletzt abgerufen am 12.04.2023.
- [112] **Grafana.** *Grafana*. <https://grafana.com/grafana/>, 2023. Zuletzt abgerufen am 12.04.2023.
- [113] **Grafana.** *Get started with Grafana and InfluxDB*. <https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-influxdb/>, 2023. Zuletzt abgerufen am 12.04.2023.
- [114] **Smartbear.** *Swagger UI*. <https://swagger.io/tools/swagger-ui/>, 2023. Zuletzt abgerufen am 01.06.2023.
- [115] **Real Games.** *Modbus TCP/IP Server*. <https://docs.factoryio.com/manual/drivers/modbus-server/>, 2023. Zuletzt abgerufen am 18.04.2023.

A. Anhang

Tabelle A.1.: Übersicht der verwendeten Tags in Factory I/O

Name	Datentyp	I/O	Task	PLC
end_sensor	bool	I	Abschliessen	%IX106.0
after_logistic_sensor	bool	I	Abschliessen	%IX106.3
logistic_pusher_conveyor	bool	O	Alle Lieferscheine	%QX109.5
pusher_sensor	bool	I	Alle Lieferscheine	
logistic_option_1_conveyor_1_3	bool	O	Express	%QX110.4
logistic_option_1_conveyor_2_3	bool	O	Express	%QX110.5
logistic_option_1_conveyor_3_3	bool	O	Express	%QX110.6
logistic_1_pusher	bool	O	Express	%QX110.2
logistic_1_pusher_front_limit	bool	I	Express	%IX105.4
logistic_1_pusher_back_limit	bool	I	Express	
logistic_1_pusher_sensor	bool	I	Express	%IX105.2
logistic_option_1_task_sensor	bool	I	Express	%IX105.5
logistic_option_1_out_sensor	bool	I	Express	%IX106.2
option_1_rfid_command	int	O	Express	
option_1_rfid_execute	bool	O	Express	
option_1_rfid_id	int	I	Express	
option_1_rfid_memoryindex	int	O	Express	
option_1_rfid_write	int	O	Express	
option_1_rfid_read	int	I	Express	
logistic_option_3_conveyor_1_4	bool	O	Grosses Paket	%QX109.6
logistic_option_3_conveyor_2_4	bool	O	Grosses Paket	%QX109.7
logistic_option_3_conveyor_3_4	bool	O	Grosses Paket	%QX110.0
logistic_option_3_conveyor_4_4	bool	O	Grosses Paket	%QX110.1
logistic_option_3_task_sensor	bool	I	Grosses Paket	%IX105.7
end_option_3_sensor	bool	I	Grosses Paket	%IX107.4
end_option_3_sensor_2	bool	I	Grosses Paket	%IX107.5
option_3_rfid_command	int	O	Grosses Paket	
option_3_rfid_execute	bool	O	Grosses Paket	
option_3_rfid_id	int	I	Grosses Paket	
option_3_rfid_memoryindex	int	O	Grosses Paket	
option_3_rfid_write	int	O	Grosses Paket	
option_3_rfid_read	int	I	Grosses Paket	
height_light_array_emitter	int	I	Hoehe	%IW106
measurement_rfid_command	int	O	Hoehe und Gewicht	
measurement_rfid_execute	bool	O	Hoehe und Gewicht	
measurement_rfid_id	int	I	Hoehe und Gewicht	
measurement_rfid_memoryindex	int	O	Hoehe und Gewicht	
measurement_rfid_write	int	O	Hoehe und Gewicht	
measurement_rfid_read	int	I	Hoehe und Gewicht	%IW111
measurement_rfid_status	int	I	Hoehe und Gewicht	
direction_stopper	bool	O	Initialisieren	%QX100.2
direction_rfid_sensor	bool	I	Initialisieren	%IX100.2
direction_rfid_command	int	O	Initialisieren	
direction_rfid_execute	bool	O	Initialisieren	
direction_rfid_id	int	I	Initialisieren	
direction_rfid_memoryindex	int	O	Initialisieren	
direction_rfid_write	int	O	Initialisieren	

fortgesetzt auf der nächsten Seite

A. Anhang

Name	Datentyp	I/O	Task	PLC
direction_rfid_read	int	I	Initialisieren	%IW108
direction_rfid_status	int	I	Initialisieren	
direction_turn	bool	O	Initialisieren	%QX100.3
direction_roll_forward	bool	O	Initialisieren	%QX100.1
direction_roll_backward	bool	O	Initialisieren	
direction_limit_0	bool	I	Initialisieren	
direction_limit_90	bool	I	Initialisieren	%IX100.3
direction_limit_back	bool	I	Initialisieren	
direction_limit_front	bool	I	Initialisieren	%IX100.1
start_rfid_sensor	bool	I	Initialisieren	%IX100.0
start_rfid_command	int	O	Initialisieren	
start_rfid_execute	bool	O	Initialisieren	
start_rfid_id	int	I	Initialisieren	
start_rfid_memoryindex	int	O	Initialisieren	
start_rfid_write	int	O	Initialisieren	
start_rfid_read	int	I	Initialisieren	
start_rfid_status	int	I	Initialisieren	
simulate_init	bool	O	Initialisieren	
simulate_extra_parts	bool	O	Kleiderhaken und Schrankstange	
extra_stuff_sensor	bool	I	Kleiderhaken und Schrankstange	
extra_stuff_rfid_command	int	O	Kleiderhaken und Schrankstange	
extra_stuff_rfid_execute	bool	O	Kleiderhaken und Schrankstange	
extra_stuff_rfid_id	int	I	Kleiderhaken und Schrankstange	
extra_stuff_rfid_memoryindex	int	O	Kleiderhaken und Schrankstange	
extra_stuff_rfid_write	int	O	Kleiderhaken und Schrankstange	
extra_stuff_rfid_read	int	I	Kleiderhaken und Schrankstange	
logistic_2_pusher_sensor	bool	I	Kleines und grosses Paket	%IX105.1
logistic_option_2_conveyor_1_1	bool	O	Kleines Paket	%QX110.7
logistic_2_pusher	bool	O	Kleines Paket	%QX110.3
logistic_2_pusher_front_limit	bool	I	Kleines Paket	%IX105.3
logistic_2_pusher_back_limit	bool	I	Kleines Paket	
logistic_option_2_task_sensor	bool	I	Kleines Paket	%IX105.6
logistic_option_2_out_sensor	bool	I	Kleines Paket	%IX106.1
option_2_rfid_command	int	O	Kleines Paket	
option_2_rfid_execute	bool	O	Kleines Paket	
option_2_rfid_id	int	I	Kleines Paket	
option_2_rfid_memoryindex	int	O	Kleines Paket	
option_2_rfid_write	int	O	Kleines Paket	
option_2_rfid_read	int	I	Kleines Paket	
after_pick_emitter	bool	O	Maschine A	%QX105.3
after_pick_conveyor_1_3	bool	O	Maschine A	%QX105.5
after_pick_conveyor_2_3	bool	O	Maschine A	%QX105.6
after_pick_conveyor_3_3	bool	O	Maschine A	%QX105.7
machining_A_trigger	bool	I	Maschine A	%IX100.6
machining_A_busy	bool	I	Maschine A	
machining_A_hasError	bool	I	Maschine A	
machining_A_produceLids	bool	O	Maschine A	
machining_A_opened	bool	I	Maschine A	
machining_A_progress	int	I	Maschine A	%IW103
machining_A_start	bool	O	Maschine A	
machining_A_stop	bool	O	Maschine A	
machining_A_reset	bool	O	Maschine A	
machining_A_remover	bool	O	Maschine A	%QX101.4
pick_sensor	bool	I	Maschine A	%IX102.7
pick_grab	bool	O	Maschine A	%QX105.0
pick_rotate_clockwise	bool	O	Maschine A	%QX105.1
pick_rotating	bool	I	Maschine A	
pick_gripper_clockwise	bool	O	Maschine A	
pick_rotate_counterclockwise	bool	O	Maschine A	%QX105.2
pick_gripper_counterclockwise	bool	O	Maschine A	

fortgesetzt auf der nächsten Seite

Name	Datentyp	I/O	Task	PLC
pick_positionX	float	I	Maschine A	%IW104
pick_positionZ	float	I	Maschine A	%IW105
pick_itemDetected	bool	I	Maschine A	%IX102.6
pick_setPositionX	float	O	Maschine A	%QW103
pick_setPositionZ	float	O	Maschine A	%QW104
pick_gripper_rotating	bool	I	Maschine A	
direction_conveyor_forward_1_3	bool	O	Maschine A	%QX100.4
direction_conveyor_forward_1_3_back	bool	O	Maschine A	
direction_conveyor_forward_1_3_left	bool	O	Maschine A	
direction_conveyor_forward_1_3_right	bool	O	Maschine A	
direction_conveyor_forward_2_3	bool	O	Maschine A	%QX100.5
direction_conveyor_forward_2_3_back	bool	O	Maschine A	
direction_conveyor_forward_2_3_left	bool	O	Maschine A	
direction_conveyor_forward_2_3_right	bool	O	Maschine A	
direction_conveyor_forward_3_3	bool	O	Maschine A	%QX101.5
direction_conveyor_forward_3_3_back	bool	O	Maschine A	
direction_conveyor_forward_3_3_left	bool	O	Maschine A	
direction_conveyor_forward_3_3_right	bool	O	Maschine A	
direction_end_sensor_forward	bool	I	Maschine A	%IX100.5
pick_chain_forward	bool	O	Maschine A	%QX105.4
pick_chain_right	bool	O	Maschine A	
pick_chain_left	bool	O	Maschine A	
pick_chain_back	bool	O	Maschine A	
pick_stopper	bool	O	Maschine A	%QX106.0
direction_forward_rfid_sensor	bool	I	Maschine A	
after_machining_a_sensor	bool	I	Maschine A	
before_machining_a_rfid_command	int	O	Maschine A	
before_machining_a_rfid_execute	bool	O	Maschine A	
before_machining_a_rfid_id	int	I	Maschine A	
before_machining_a_rfid_memoryindex	int	O	Maschine A	
before_machining_a_rfid_write	int	O	Maschine A	
before_machining_a_rfid_read	int	I	Maschine A	
after_machining_a_rfid_command	int	O	Maschine A	
after_machining_a_rfid_execute	bool	O	Maschine A	
after_machining_a_rfid_id	int	I	Maschine A	
after_machining_a_rfid_memoryindex	int	O	Maschine A	
after_machining_a_rfid_write	int	O	Maschine A	
after_machining_a_rfid_read	int	I	Maschine A	
pick_stopper	bool	O	Maschine A	
machining_B_remover_sensor	bool	I	Maschine B	%IX100.7
machining_B_remover	bool	O	Maschine B	%QX101.0
machining_B_trigger	bool	I	Maschine B	%IX101.2
machining_B_busy	bool	I	Maschine B	
machining_B_hasError	bool	I	Maschine B	
machining_B_produceLids	bool	O	Maschine B	
machining_B_opened	bool	I	Maschine B	
machining_B_progress	int	I	Maschine B	%IW100
machining_B_start	bool	O	Maschine B	
machining_B_stop	bool	O	Maschine B	
machining_B_reset	bool	O	Maschine B	
spawn_pallett_trigger_sensor	bool	I	Maschine B	%IX101.0
spawn_pallett_emit	bool	O	Maschine B	%QX100.7
spawn_pallett_conveyor	bool	O	Maschine B	%QX100.6
spawn_pallett_stopper	bool	O	Maschine B	%QX101.6
spawn_pallett_sensor	bool	I	Maschine B	%IX101.3
spawn_pallet_end_sensor	bool	I	Maschine B	%IX101.1
spawn_curve_1_2	bool	O	Vorbohren (Schrack)	%QX101.7
spawn_curve_2_2	bool	O	Regale	%QX102.0
direction_conveyor_left	bool	O	Maschine B	%QX101.1
direction_conveyor_left_1_2	bool	O	Maschine B	%QX101.2

fortgesetzt auf der nächsten Seite

A. Anhang

Name	Datentyp	I/O	Task	PLC
direction_conveyor_left_1_2_back	bool	O	Maschine B	%QX101.3
direction_conveyor_left_1_2_left	bool	O	Maschine B	
direction_conveyor_left_1_2_right	bool	O	Maschine B	
direction_conveyor_left_2_2	bool	O	Maschine B	
direction_conveyor_left_2_2_back	bool	O	Maschine B	
direction_conveyor_left_2_2_left	bool	O	Maschine B	
direction_conveyor_left_2_2_right	bool	O	Maschine B	%IX100.4
direction_end_sensor_left	bool	I	Maschine B	
after_machining_b_rfid_command	int	O	Maschine B	
after_machining_b_rfid_execute	bool	O	Maschine B	
after_machining_b_rfid_id	int	I	Maschine B	
after_machining_b_rfid_memoryindex	int	O	Maschine B	
after_machining_b_rfid_write	int	O	Maschine B	%IX106.4
after_machining_b_rfid_read	int	I	Maschine B	
before_machining_b_rfid_command	int	O	Maschine B	
before_machining_b_rfid_execute	bool	O	Maschine B	
before_machining_b_rfid_id	int	I	Maschine B	
before_machining_b_rfid_memoryindex	int	O	Maschine B	
before_machining_b_rfid_write	int	O	Maschine B	%IX106.4
before_machining_b_rfid_read	int	I	Maschine B	
spawn_pallett_rfid_command	int	O	Maschine B	
spawn_pallett_rfid_execute	bool	O	Maschine B	
spawn_pallett_rfid_id	int	I	Maschine B	
spawn_pallett_rfid_memoryindex	int	O	Maschine B	
spawn_pallett_rfid_write	int	O	Maschine B	%IX106.4
spawn_pallett_rfid_read	int	I	Maschine B	
after_varnishing_conveyor	bool	O	Oberflaechenbehandlung	
cross_sensor	bool	I	Oberflaechenbehandlung	
join_chain_forward	bool	O	Oberflaechenbehandlung	
join_chain_back	bool	O	Oberflaechenbehandlung	
join_chain_left	bool	O	Oberflaechenbehandlung	
join_chain_right	bool	O	Oberflaechenbehandlung	%IX106.4
before_cross_left_conveyor	bool	O	Oberflaechenbehandlung	
before_varnishing_conveyor_1_2	bool	O	Oberflaechenbehandlung	
before_varnishing_conveyor_2_2	bool	O	Oberflaechenbehandlung	
varnishing_stopper_1_2	bool	O	Oberflaechenbehandlung	
varnishing_stopper_2_2	bool	O	Oberflaechenbehandlung	
varnishing_trigger_sensor	bool	I	Oberflaechenbehandlung	%IX106.4
varnishing_end_sensor	bool	I	Oberflaechenbehandlung	
varnishing_rfid_sensor	bool	I	Oberflaechenbehandlung	
varnishing_rfid_command	int	O	Oberflaechenbehandlung	
varnishing_rfid_execute	bool	O	Oberflaechenbehandlung	
varnishing_rfid_id	int	I	Oberflaechenbehandlung	
varnishing_rfid_memoryindex	int	O	Oberflaechenbehandlung	%IX106.4
varnishing_rfid_write	int	O	Oberflaechenbehandlung	
varnishing_rfid_read	int	I	Oberflaechenbehandlung	
varnishing_rfid_status	int	I	Oberflaechenbehandlung	
varnishing_conveyor	bool	O	Oberflaechenbehandlung	
varnishing_start_sensor	bool	I	Oberflaechenbehandlung	
tank_fillValve	float	I	Oberflaechenbehandlung	%IX106.4
tank_flowMeter	float	I	Oberflaechenbehandlung	
tank_levelMeter	float	O	Oberflaechenbehandlung	
tank_dischargeValve	float	O	Oberflaechenbehandlung	
before_cross_left_stopper	bool	O	Oberflaechenbehandlung	
before_cross_right_stopper	bool	O	Oberflaechenbehandlung	
cross_center_sensor	bool	I	Oberflaechenbehandlung	%IX106.4
join_enter_left_sensor	bool	I	Oberflaechenbehandlung	
join_enter_right_sensor	bool	I	Oberflaechenbehandlung	
palletizer_chain_forward	bool	O	Palette beladen	
palletizer_chain_back	bool	O	Palette beladen	

fortgesetzt auf der nächsten Seite

Name	Datentyp	I/O	Task	PLC
palletizer_belt_forward	bool	O	Palette beladen	
palletizer_belt_back	bool	O	Palette beladen	%QX112.5
palletizer_openPlate	bool	O	Palette beladen	%QX112.7
palletizer_plateLimit	bool	I	Palette beladen	%IX107.3
palletizer_clamp	bool	O	Palette beladen	%QX112.0
palletizer_clamped	bool	I	Palette beladen	%IX106.6
palletizer_pusherLimit	bool	I	Palette beladen	%IX107.6
palletizer_push	bool	O	Palette beladen	%QX112.6
palletizer_turn	bool	O	Palette beladen	
palletizer_elevatorMoving	bool	I	Palette beladen	%IX106.7
palletizer_moveToLimit	bool	O	Palette beladen	%QX112.3
palletizer_elevator_up	bool	O	Palette beladen	%QX112.2
palletizer_elevator_down	bool	O	Palette beladen	%QX112.1
palletizer_elevator_frontLimit	bool	I	Palette beladen	%IX107.0
palletizer_elevator_backLimit	bool	I	Palette beladen	
before_pack_conveyor_1_3	bool	O	Palette beladen	%QX108.4
before_pack_conveyor_2_3	bool	O	Palette beladen	%QX108.5
before_pack_conveyor_3_3	bool	O	Palette beladen	%QX108.6
pre_palletizer_emitter	bool	O	Palette beladen	%QX111.6
spawn_pallet_trigger_sensor_2	bool	I	Palette beladen	%IX107.1
palletizer_rfid_command	int	O	Palette beladen	
palletizer_rfid_execute	bool	O	Palette beladen	
palletizer_rfid_id	int	I	Palette beladen	
palletizer_rfid_memoryindex	int	O	Palette beladen	
palletizer_rfid_write	int	O	Palette beladen	
palletizer_rfid_read	int	I	Palette beladen	
pallet_sensor	bool	I	Palette beladen	
pallet_end_sensor	bool	I	Palette beladen	
pre_palletizer_conveyor	bool	O	Palletizer	
check_rfid_sensor	bool	I	Qualitaet Pruefen	%IX104.2
before_check_direction_sensor	bool	I	Qualitaet Pruefen	%IX104.3
check_rfid_command	int	O	Qualitaet Pruefen	
check_rfid_execute	bool	O	Qualitaet Pruefen	
check_rfid_id	int	I	Qualitaet Pruefen	
check_rfid_memoryindex	int	O	Qualitaet Pruefen	
check_rfid_write	int	O	Qualitaet Pruefen	
check_rfid_read	int	I	Qualitaet Pruefen	%IW109
check_rfid_status	int	I	Qualitaet Pruefen	
check_out_sensor	bool	I	Qualitaet Pruefen	%IX104.0
repeat_out_sensor	bool	I	Qualitaet Pruefen	%IX104.1
check_direction_roll_forward	bool	O	Qualitaet Pruefen	%QX108.2
check_direction_roll_back	bool	O	Qualitaet Pruefen	%QX108.3
check_direction_turn	bool	O	Qualitaet Pruefen	%QX108.1
check_direction_limit_front	bool	I	Qualitaet Pruefen	%IX103.5
check_direction_limit_back	bool	I	Qualitaet Pruefen	%IX103.4
check_direction_limit_90	bool	I	Qualitaet Pruefen	%IX103.7
check_direction_limit_0	bool	I	Qualitaet Pruefen	%IX103.6
before_quality_conveyor_1_2	bool	O	Qualitaet Pruefen	%QX107.4
before_check_stopper	bool	O	Qualitaet Pruefen	%QX107.2
simulate_quality_check	bool	O	Qualitaet Pruefen	
after_quality_conveyor_3_3	bool	O	Qualitaet verbessern	%QX108.0
before_quality_conveyor_2_2	bool	O	Qualitaet verbessern	%QX107.5
after_quality_conveyor_2_3	bool	O	Qualitaet verbessern	%QX107.7
after_quality_conveyor_1_3	bool	O	Qualitaet verbessern	%QX107.6
quality_stopper	bool	O	Qualitaet verbessern	
improve_sensor	bool	I	Qualitaet verbessern	%IX107.7
improve_rfid_command	int	O	Qualitaet verbessern	
improve_rfid_execute	bool	O	Qualitaet verbessern	
improve_rfid_id	int	I	Qualitaet verbessern	
improve_rfid_memoryindex	int	O	Qualitaet verbessern	

fortgesetzt auf der nächsten Seite

A. Anhang

Name	Datentyp	I/O	Task	PLC
improve_rfid_write	int	O	Qualitaet verbessern	
improve_rfid_read	int	I	Qualitaet verbessern	
simulate_improve_quality	bool	O	Qualitaet verbessern	
shelves_rfid_command	int	O	Regale	
shelves_rfid_execute	bool	O	Regale	
shelves_rfid_id	int	I	Regale	
shelves_rfid_memoryindex	int	O	Regale	
shelves_rfid_write	int	O	Regale	
shelves_rfid_read	int	I	Regale	
simulate_shelves_production	bool	O	Regale	
simulate_drawers_production	bool	O	Schubladen	
drawers_sensor	bool	I	Schubladen	
drawers_rfid_command	int	O	Schubladen	
drawers_rfid_execute	bool	O	Schubladen	
drawers_rfid_id	int	I	Schubladen	
drawers_rfid_memoryindex	int	O	Schubladen	
drawers_rfid_write	int	O	Schubladen	
drawers_rfid_read	int	I	Schubladen	
elevator_1_forward	bool	O	Spiegel und Tuer	%QX102.1
elevator_1_back	bool	O	Spiegel und Tuer	
elevator_1_left_limit	bool	I	Spiegel und Tuer	%IX101.5
elevator_1_right_limit	bool	I	Spiegel und Tuer	%IX101.4
elevator_1_set_point	float	O	Spiegel und Tuer	%QW101
elevator_1_position	float	I	Spiegel und Tuer	%IW101
first_floor_trigger_sensor	bool	I	Spiegel vorbereiten	%IX101.6
second_floor_trigger_sensor	bool	I	Spiegel vorbereiten	%IX102.4
second_floor_rfid_command	int	O	Spiegel vorbereiten	
second_floor_rfid_execute	bool	O	Spiegel vorbereiten	
second_floor_rfid_id	int	I	Spiegel vorbereiten	
second_floor_rfid_memoryindex	int	O	Spiegel vorbereiten	
second_floor_rfid_write	int	O	Spiegel vorbereiten	
second_floor_rfid_read	int	I	Spiegel vorbereiten	
second_floor_rfid_status	int	I	Spiegel vorbereiten	
second_floor_emitter	bool	O	Spiegel vorbereiten	%QX104.5
elevator_1_rfid_command	int	O	Spiegel vorbereiten	
elevator_1_rfid_execute	bool	O	Spiegel vorbereiten	
elevator_1_rfid_id	int	I	Spiegel vorbereiten	
elevator_1_rfid_memoryindex	int	O	Spiegel vorbereiten	
elevator_1_rfid_write	int	O	Spiegel vorbereiten	
elevator_1_rfid_read	int	I	Spiegel vorbereiten	
simulate_prepare_mirror_material	bool	O	Spiegel vorbereiten	
second_floor_conveyor_1_8	bool	O	Spiegel zuschneiden	%QX103.5
second_floor_conveyor_2_8	bool	O	Spiegel zuschneiden	%QX103.6
second_floor_conveyor_3_8	bool	O	Spiegel zuschneiden	%QX103.7
second_floor_conveyor_4_8	bool	O	Spiegel zuschneiden	%QX104.0
second_floor_conveyor_5_8	bool	O	Spiegel zuschneiden	%QX104.1
second_floor_conveyor_6_8	bool	O	Spiegel zuschneiden	%QX104.2
second_floor_conveyor_7_8	bool	O	Spiegel zuschneiden	%QX104.3
second_floor_conveyor_8_8	bool	O	Spiegel zuschneiden	%QX104.4
shape_sensor	bool	I	Spiegel zuschneiden	
shape_rfid_command	int	O	Spiegel zuschneiden	
shape_rfid_execute	bool	O	Spiegel zuschneiden	
shape_rfid_id	int	I	Spiegel zuschneiden	
shape_rfid_memoryindex	int	O	Spiegel zuschneiden	
shape_rfid_write	int	O	Spiegel zuschneiden	
shape_rfid_read	int	I	Spiegel zuschneiden	
angular	bool	O	Spiegel zuschneiden	
circular	bool	O	Spiegel zuschneiden	
start_conveyor	bool	O	Start	%QX100.0
start_emitter	bool	O	Start	

fortgesetzt auf der nächsten Seite

Name	Datentyp	I/O	Task	PLC
start_emitter_part	int	O	Start	
start_emitter_base	int	O	Start	%QW109
start_counter	int?	O	Start	%QW100
start_sensor	bool	I	Start	%IX106.5
table_style_rfid_command	int	O	Tisch Stil	
table_style_rfid_execute	bool	O	Tisch Stil	
table_style_rfid_id	int	I	Tisch Stil	
table_style_rfid_memoryindex	int	O	Tisch Stil	
table_style_rfid_write	int	O	Tisch Stil	
table_style_rfid_read	int	I	Tisch Stil	
table_style_rfid_status	int	I	Tisch Stil	
classic	bool	O	Tisch Stil	
modern	bool	O	Tisch Stil	
table_style_sensor	bool	I	Tisch Stil	
simulate_table_legs_production	bool	O	Tischbeine	
table_legs_sensor	bool	I	Tischbeine	
table_legs_rfid_command	int	O	Tischbeine	
table_legs_rfid_execute	bool	O	Tischbeine	
table_legs_rfid_id	int	I	Tischbeine	
table_legs_rfid_memoryindex	int	O	Tischbeine	
table_legs_rfid_write	int	O	Tischbeine	
table_legs_rfid_read	int	I	Tischbeine	
first_floor_conveyor_1_8	bool	O	Tuer Typ	%QX102.2
first_floor_conveyor_2_8	bool	O	Tuer Typ	%QX102.3
first_floor_conveyor_3_8	bool	O	Tuer Typ	%QX102.4
first_floor_conveyor_4_8	bool	O	Tuer Typ	%QX102.5
slidingDoor	bool	O	Tuer Typ (Schiebetuer)	
defaultDoor	bool	O	Tuer Typ (Default)	
door_type_sensor	bool	I	Tuer Typ	
first_floor_conveyor_5_8	bool	O	Tuer Typ und Tuerschloss	%QX102.6
first_floor_conveyor_6_8	bool	O	Tuerschloss	%QX102.7
first_floor_conveyor_7_8	bool	O	Tuerschloss	%QX103.0
first_floor_conveyor_8_8	bool	O	Tuerschloss	%QX103.1
lock_rfid_command	int	O	Tuerschloss	
lock_rfid_execute	bool	O	Tuerschloss	
lock_rfid_id	int	I	Tuerschloss	
lock_rfid_memoryindex	int	O	Tuerschloss	
lock_rfid_write	int	O	Tuerschloss	
lock_rfid_read	int	I	Tuerschloss	
lock_sensor	bool	I	Tuerschloss	
simulate_lock_production	bool	O	Tuerschloss	
door_type_rfid_command	int	O	Tuertyp	
door_type_rfid_execute	bool	O	Tuertyp	
door_type_rfid_id	int	I	Tuertyp	
door_type_rfid_memoryindex	int	O	Tuertyp	
door_type_rfid_write	int	O	Tuertyp	
door_type_rfid_read	int	I	Tuertyp	
door_type_rfid_status	int	I	Tuertyp	
after_check_stoppper	bool	O	Verbessern	%QX107.3
pack_start_sensor	bool	I	Verpacken	%IX104.4
pack_emitter	bool	O	Verpacken	%QX109.3
pack_emitter_base	int	O	Verpacken	%QW105
pack_emitter_part	int	O	Verpacken	%QW106
pack_remover	bool	O	Verpacken	%QX109.2
pack_trigger_sensor	bool	I	Verpacken	%IX104.7
pack_end_sensor	bool	I	Verpacken	%IX104.6
pack_conveyor	bool	O	Verpacken	%QX108.7
after_pack_conveyor	bool	O	Verpacken	%QX109.1
pack_rfid_sensor	bool	I	Verpacken	%IX104.5
pack_rfid_command	int	O	Verpacken	

fortgesetzt auf der nächsten Seite

A. Anhang

Name	Datentyp	I/O	Task	PLC
pack_rfid_execute	bool	O	Verpacken	
pack_rfid_id	int	I	Verpacken	
pack_rfid_memoryindex	int	O	Verpacken	
pack_rfid_write	int	O	Verpacken	
pack_rfid_read	int	I	Verpacken	
pack_rfid_status	int	I	Verpacken	
pack_stopper	bool	O	Verpacken	%QX109.0
after_pack_sensor	bool	I	Verpacken	%IX105.0
box_rfid_command	int	O	Verpacken	
box_rfid_execute	bool	O	Verpacken	
box_rfid_id	int	I	Verpacken	
box_rfid_memoryindex	int	O	Verpacken	
box_rfid_write	int	O	Verpacken	
box_rfid_read	int	I	Verpacken	%IW111
after_logistic_conveyor	bool	O	Versenden	%QX111.1
after_logistic_remover	bool	O	Versenden	
after_palletizer_conveyor	bool	O	Versenden	%QX111.4
after_palletizer_remover	bool	O	Versenden	
end_sensor_2	bool	I	Versenden	%IX107.2
product_shipment_rfid_command	int	O	Versenden	
product_shipment_rfid_execute	bool	O	Versenden	
product_shipment_rfid_id	int	I	Versenden	
product_shipment_rfid_memoryindex	int	O	Versenden	
product_shipment_rfid_write	int	O	Versenden	
product_shipment_rfid_read	int	I	Versenden	
pallet_shipment_rfid_command	int	O	Versenden	
pallet_shipment_rfid_execute	bool	O	Versenden	
pallet_shipment_rfid_id	int	I	Versenden	
pallet_shipment_rfid_memoryindex	int	O	Versenden	
pallet_shipment_rfid_write	int	O	Versenden	
pallet_shipment_rfid_read	int	I	Versenden	
after_machining_b_sensor	bool	I	Vorbohren (Schrank)	
simulate_pre_drill_schrank	bool	O	Vorbohren (Schrank)	
pre_drill_b_sensor	bool	I	Vorbohren (Schrank)	
pre_drill_b_rfid_command	int	O	Vorbohren (Schrank)	
pre_drill_b_rfid_execute	bool	O	Vorbohren (Schrank)	
pre_drill_b_rfid_id	int	I	Vorbohren (Schrank)	
pre_drill_b_rfid_memoryindex	int	O	Vorbohren (Schrank)	
pre_drill_b_rfid_write	int	O	Vorbohren (Schrank)	
pre_drill_b_rfid_read	int	I	Vorbohren (Schrank)	
simulate_pre_drill	bool	O	Vorbohren (Schreibtisch)	
pre_drill_a_sensor	bool	I	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_command	int	O	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_execute	bool	O	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_id	int	I	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_memoryindex	int	O	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_write	int	O	Vorbohren (Schreibtisch)	
pre_drill_a_rfid_read	int	I	Vorbohren (Schreibtisch)	
scale_forward	bool	O	Waage	%QX109.4
scale_back	bool	O	Waage	
scale_weight	float	I	Waage	
first_floor_end_sensor	bool	I	Zusammenbauen	%IX101.7
first_floor_stopper	bool	O	Zusammenbauen	%QX103.2
second_floor_end_sensor	bool	I	Zusammenbauen	%IX102.5
second_floor_remover	bool	O	Zusammenbauen	%QX104.6
second_floor_stopper	bool	O	Zusammenbauen	%QX104.7
elevator_2_forward	bool	O	Zusammenbauen	%QX103.3
elevator_2_back	bool	O	Zusammenbauen	
elevator_2_left_limit	bool	I	Zusammenbauen	%IX102.1
elevator_2_right_limit	bool	I	Zusammenbauen	%IX102.0

fortgesetzt auf der nächsten Seite

Name	Datentyp	I/O	Task	PLC
elevator_2_end_sensor	bool	I	Zusammenbauen	%IX102.2
elevator_2_set_point	float	O	Zusammenbauen	%QW102
elevator_2_position	float	I	Zusammenbauen	%IW102
simulate_assemble	bool	O	Zusammenbauen	
assemble_rfid_command	int	O	Zusammenbauen	
assemble_rfid_execute	bool	O	Zusammenbauen	
assemble_rfid_id	int	I	Zusammenbauen	
assemble_rfid_memoryindex	int	O	Zusammenbauen	
assemble_rfid_write	int	O	Zusammenbauen	
assemble_rfid_read	int	I	Zusammenbauen	
additional_equipment_sensor	bool	I	Zusatzmaterial	
additional_equipment_rfid_command	int	O	Zusatzmaterial	
additional_equipment_rfid_execute	bool	O	Zusatzmaterial	
additional_equipment_rfid_id	int	I	Zusatzmaterial	
additional_equipment_rfid_memoryindex	int	O	Zusatzmaterial	
additional_equipment_rfid_write	int	O	Zusatzmaterial	
additional_equipment_rfid_read	int	I	Zusatzmaterial	
simulate_additional_equipment	int	O	Zusatzmaterial	

Tabelle A.2.: Petri-Netz Bezeichnungen Anwendungsfall 1: Produktion von Möbel

Element	Bezeichnung
p1	Bestellung erhalten
t1	Material auf Palette vorbereiten
p2	Material auf Palette vorbereitet
t2	RFID Transponder initialisieren
p3	RFID Transponder initialisiert
t3	Produktart in Bestelldetails abfragen
p4	Schrank ausgewählt
t4	Material zuschneiden mit Maschine A
p5	Material zugeschnitten
t5	Löcher vorbohren
p6	Löcher vorgebohrt
t6	Regale herstellen
p7	Regale hergestellt
t7	Bestandteile herstellen
p8	Bereit Tür herzustellen
t8	Art der Tür in Bestelldetails abfragen
p9	Schiebetür ausgewählt
t9	Schiebetür herstellen
p10	Schiebetür hergestellt
p11	Standardtür ausgewählt
t10	Standardtür herstellen
p12	Standardtür hergestellt
t11	Türherstellung abschließen
p13	Türherstellung abgeschlossen
t12	Türschloss einbauen
p14	Türschloss eingebaut
t19	Bestandteile verbauen
p22	Bestandteile verbaut
p15	Bereit Spiegel zu bearbeiten
t14	Spiegel Material vorbereiten
p16	Spiegel Material vorbereitet
t15	Spiegelform in Bestelldetails abfragen
p17	Rund ausgewählt
t16	Spiegel rund zuschneiden
p18	Spiegel rund zugeschnitten
p19	Eckig ausgewählt

fortgesetzt auf der nächsten Seite

A. Anhang

Element	Bezeichnung
t17	Spiegel eckig zuschneiden
p20	Spiegel eckig zugeschnitten
t18	Spiegel fürs Anbringen vorbereiten
p21	Spiegel bereits fürs Anbringen
t20	Kleiderhaken und Schrankstange beilegen
p23	Kleiderhaken und Schrankstange beigelegt
p24	Schreibtisch ausgewählt
t19	Material zuschneiden mit Maschine B
p25	Material zugeschnitten
t20	Löcher vorbohren
p26	Löcher vorgebohrt
t21	Schubladen herstellen
p27	Schubladen hergestellt
t22	Stil des Tisches abfragen
p28	Klassisch ausgewählt
t23	Tischplatte gravieren
p29	Tischplatte graviert
p30	Modern ausgewählt
t24	Kabeldurchführung einfügen
p31	Kabeldurchführung eingefügt
t25	Ausgewählte Verarbeitungsmethode abschließen
p32	Verarbeitungsmethode abgeschlossen
t26	Tischbeine herstellen
p33	Tischbeine hergestellt
t27	Produkt für Oberflächenbehandlung vorbereiten
p34	Produkt für Oberflächenbehandlung vorbereitet
t28	Oberflächenbehandlung durchführen
p35	Oberflächenbehandlung durchgeführt
t29	Qualitätskontrolle durchführen
p36	Qualitätskontrolle durchgeführt
t30	Produkt nicht vollständig
p37	Fehlende Bestandteile beigelegt
t31	Produkt vollständig

Tabelle A.3.: Petri-Netz Bezeichnungen Anwendungsfall 2: Verpackung und Versand

Element	Bezeichnung
p1	Produkt bereit zum Versand
t1	Zusatzausrüstung in Bestelldetails abfragen
p2	Zusatzausrüstung erwünscht
t2	Zusatzausrüstung beilegen
p3	Zusatzausrüstung beigelegt
p4	Keine Zusatzausrüstung erforderlich
t3	Produkt fürs Verpacken vorbereiten
p5	Produkt fürs Verpacken vorbereitet
t4	Passendes Paket ermitteln
p6	Schreibtisch und keine Zusatzausrüstung erforderlich
t5	In Box S verpacken
p7	In Box S verpackt
p8	Schreibtisch und Zusatzausrüstung bzw. Schreibtisch und keine Zusatzausrüstung erforderlich
t6	In Box M verpacken
p9	In Box M verpackt
p10	Schreibtisch und Zusatzausrüstung erforderlich
t7	In Box L verpacken
p11	In Box L verpackt
t8	Verpacken abschließen
p12	Verpacken abgeschlossen
t9	Höhe des Paketes messen

fortgesetzt auf der nächsten Seite

Element	Bezeichnung
p13	Höhe des Paketes gemessen
t10	Paket wiegen
p14	Gewicht des Paketes ermittelt
t11	Paketdimensionen kategorisieren
p15	Höhe $\leq$ X und Gewicht $\leq$ Y und keine Expresslieferung
t12	Lieferschein für kleines Paket drucken
p16	Lieferschein für kleines Paket gedruckt
p17	Höhe $>$ X und Gewicht $>$ Y und keine Expresslieferung
t13	Lieferschein für großes Paket drucken
p18	Lieferschein für großes Paket gedruckt
t14	Paket auf Palette stapeln
p19	Paket auf Palette gestapelt
t15	Warten bis Palette voll beladen ist
p20	Palette wurde voll beladen
p21	Expresslieferung
t16	Expresslieferschein drucken
p22	Expresslieferschein gedruckt
t17	Produkt/Palette zum Versand vorbereiten
p23	Produkt/Palette zum Versand vorbereitet
t18	Paket bzw. Palette versenden
p24	Paket bzw. Palette versendet