Die Virtuelle Klemmleiste – ein Ansatz zur Integration der Verhaltenslogik in den Digitalen Zwilling

Vom Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur
Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

DISSERTATION

vorgelegt von

Vladimir Kutscher, M.Sc.

geb. Rezanov

aus Schanatas, Kasachstan

Berichterstatter:	Prof. Dr.-Ing. Reiner Anderl
Mitberichterstatter:	Prof. Dr. rer. nat. Andy Schürr
Tag der Einreichung:	27.04.2023
Tag der mündlichen Prüfung:	12.07.2023

Darmstadt 2023
D17

Kutscher, Vladimir : Die Virtuelle Klemmleiste – ein Ansatz zur Integration der Verhaltenslogik in den Digitalen Zwilling
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2023
URN: urn:nbn:de:tuda-tuprints-247075
Tag der mündlichen Prüfung: 12.07.2023

Geleitwort des Herausgebers

Die moderne Informations- und Kommunikationstechnologie bietet vielfältige Innovations- und Leistungspotentiale, die im Entstehungsprozess neuer Produkte auszuschöpfen sind. Dies setzt jedoch voraus, dass die wissenschaftlichen Grundlagen zum Einsatz der modernen IKT in der Produktentstehung vorliegen und neue Methoden wissenschaftlich abgesichert sind. Darüber hinaus stellen die wissenschaftliche Durchdringung und die Bereitstellung wissenschaftlicher Forschungsergebnisse eine abgestimmte Kooperation zwischen Forschung und Industrie dar.

Ziel der Forschungsarbeiten ist die wissenschaftliche Durchdringung innovativer, interdisziplinärer und integrierter Produktentstehungsprozesse und darauf aufbauend die Konzeption neuer Methoden für die Entwicklung, Konstruktion, Arbeitsvorbereitung und Herstellung neuer Produkte.

Solche neuartigen Produkte bilden Cyber-Physische Systeme, welche physischen Prozesse mit Datenverarbeitung verbinden und dabei über multimodale Schnittstellen mit der Umwelt, weiteren technischen Systemen und dem Menschen interagieren können. Die Entwicklung dieser Systeme stellt Ingenieure vor neue Herausforderungen.

Zur Unterstützung der Entwicklung und des Betriebs dieser Systemart werden Digitalen Zwillinge eingesetzt, welche zum Ziel haben, sowohl die physische als auch die digitale (cyber) Seite der Cyber-Physischen Systeme virtuell zu repräsentieren. Eine besondere Problemstellung erwächst dabei aus der Verbindung zwischen der Hardware und Software, welche das Verhalten maßgeblich bestimmt.

Herr Kutscher adressiert diese Problemstellung mittels der vorliegenden Dissertation und entwickelt ein Konzept zur Integration der Verhaltenslogik eines Cyber-Physischen Systems in dessen Digitalen Zwilling. Das Konzept sieht die Virtualisierung der Verhaltenslogik mittels Simulation der eingebetteten Elektronik vor. Den wissenschaftlichen Kern stellt die als Virtuelle Klemmleiste bezeichnete Schnittstelle dar, die virtualisierte Verhaltenslogik mit weiteren Komponenten des Digitalen Zwillings eines Cyber-Physischen Systems verknüpft.

Im Oktober 2023 *Prof. Dr.-Ing. Reiner Anderl*

Vorwort des Autors

Die vorliegende Dissertation entstand hauptsächlich während meiner Tätigkeit als Wissenschaftlicher Mitarbeiter am Fachgebiet Datenverarbeitung in der Konstruktion (DiK) der Technischen Universität Darmstadt. Im letzten Tätigkeitsjahr wurde dieses im Rahmen des Wechsels der Institutsleitung zum Fachgebiet Product Life Cycle Management (PLCM).

Mein besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. Reiner Anderl, Leiter des Fachgebiets DiK. Seine wertvollen Anregungen und die fachlichen Diskussionen sowie die Ermöglichung wissenschaftlicher Freiräume zur Entfaltung eigener Ideen haben einen wichtigen Beitrag zur Entstehung dieser Dissertation geleistet.

Ich bedanke mich bei allen ehemaligen und aktuellen Kolleginnen und Kollegen des Fachgebiets für die freundliche und kreative Arbeitsatmosphäre und die zahlreichen fachlichen Impulse, die einen wesentlichen Anteil zum Gelingen dieser Dissertation beigetragen haben. Besonderer Dank gilt Herrn Johannes Olbort, Herrn Dr.-Ing. Erdal Tantik, Herrn Dr.-Ing. Oleg Anokhin, Herrn Dr.-Ing. Thomas Dasbach, Herrn Benjamin Röhm, Herrn Slim Krückemeier und Herrn Christian Plesker für fachliche Diskussionen sowie für die Durchsicht meiner Dissertation.

Ganz herzlich möchte ich danken meiner Familie und insbesondere meiner Frau Jennifer, die mich auf meinem Weg zur Dissertation stets unterstützt und motiviert hat. Sie war mein Rückhalt in schwierigen Zeiten und zugleich eine Quelle der Freude und Inspiration. Meiner Familie ist diese Dissertation gewidmet.

Michelstadt, März 2023 *Vladimir Kutscher*

Zusammenfassung

Die Vision eines Digitalen Zwillings beschreibt ein durchgängiges Abbild eines Systems, welches in diversen Anwendungsfällen als virtuelle Repräsentation eingesetzt werden kann und in einer bidirektionalen Verbindung mit dem Physischen Zwilling steht. Der Digitale Zwilling soll dabei Eigenschaften der Physischen Zwillings abbilden. Zu den wesentlichen Eigenschaften eines mechatronischen bzw. Cyber-Physischen Systems gehört auch das inhärente Verhalten, welches auf der individuellen Verhaltenslogik des Systems basiert. Die Verhaltenslogik wird durch die eingebettete Software und Elektronik implementiert. Die Integration der originalgetreuen Verhaltenslogik, welche aus der Elektronik und der eingebetteten Software des Physischen Zwillings hervorgeht, ist wissenschaftlich bisher nicht durchdrungen.

Aus dieser Forschungslücke leitet sich die Zielsetzung der Dissertation ab. Das Ziel der Dissertation ist die Integration der Verhaltenslogik in den Digitalen Zwilling. Im Fokus liegt dabei die Nutzung der originalen eingebetteten Software, die mittels einer virtuellen Abbildung der Elektronik zum direkten Einsatz im Digitalen Zwilling befähigt wird. Zur Erreichung des Ziels erfolgt im Kern der Dissertation die Entwicklung eines Konzepts einer Virtuellen Klemmleiste, welche die Verbindung zwischen der virtuellen Verhaltenslogik und den anschließenden Partialmodellen des Digitalen Zwillings herstellt und dadurch die Integration des Verhaltens in den Digitalen Zwilling ermöglicht. Der zugehörige Prozess der Integration der Verhaltenslogik in den Digitalen Zwilling besteht aus den zwei Teilprozessen der *Virtualisierung der Verhaltenslogik* und der *Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik*.

Zur Validierung der Tragfähigkeit des Konzepts wird der Prozess am Beispiel eines CNC (*Computerized Numerical Control*) Laserplotters durchgeführt und implementiert. Die Validierung zeigt, dass das entwickelte Konzept geeignet ist, um das originalgetreue Verhalten in den Digitalen Zwilling zu integrieren.

Abstract

The vision of a digital twin describes a consistent copy of a system that can be used as a virtual representation in various use cases and has a bidirectional connection with the physical twin. The digital twin should represent properties of the physical twin. One of the essential properties of a mechatronic or cyber-physical system is its inherent behavior, which is based on the individual behavioral logic of the system. In this context, the behavioral logic is implemented by the embedded software and electronics. The integration of the original behavioral logic, which emerges from the electronics and embedded software of the physical twin, has not yet been scientifically investigated.

The objective of the dissertation is derived from this research gap. The goal of the dissertation is the integration of behavioral logic into the digital twin. The focus is on the use of the original embedded software, which is enabled for direct use in the digital twin by means of a virtual representation of the electronics. To achieve the goal, a concept was developed that includes the description of the process of integrating the behavior based on and the development of an information model-based interface. The integration of behavioral logic into the digital twin consists of the two sub-processes of *virtualization of behavioral logic* and *extension of a digital twin by virtualized behavioral logic*. In addition, a *virtual terminal* is developed, which establishes the connection between the virtual behavioral logic and the subsequent partial models of the digital twin.

To validate the concept's viability, the process is carried out and implemented using a CNC (*Computerized Numerical Control*) laser plotter as an example. The validation shows that the developed concept is suitable for integrating the original behavior into the digital twin.

Inhaltsverzeichnis

Geleitwort des Herausgebers **iii**

Vorwort des Autors **v**

Abbildungsverzeichnis **xv**

Tabellenverzeichnis **xix**

Abkürzungsverzeichnis **xxi**

1 Einleitung **1**
1.1 Motivation und Problemstellung 1
1.2 Zielsetzung 3
1.3 Aufbau der Dissertation 5

2 Stand der Technik und Forschung **7**
2.1 Virtuelle Produktentwicklung 7
2.1.1 Produktentwicklungsprozess 8
2.1.2 Entwicklung eingebetteter Software 10
2.1.3 Entwicklung der Elektrotechnik und Elektronik 13
2.2 Digitalisierung und Industrie 4.0 14
2.2.1 Cyber-Physisches System 14
2.2.2 Verwaltungsschale 17
2.3 Digitaler Zwilling 18
2.3.1 Einleitung und Definition 19
2.3.2 Charakteristika 21
2.3.3 Architektur und Schnittstellen 25
2.3.4 Design 27
2.3.5 Verhaltensabbildung 28
2.3.6 Bereitstellung Digitaler Zwillinge 31

2.4 Simulation im Digitalen Zwilling . . . 32
2.4.1 Co-Simulation . . . 33
2.4.2 Architektur- und Schnittstellenstandards . . . 37
2.4.3 Virtuelle Plattformen . . . 41
2.5 Fazit zum Stand der Technik . . . 44

3 Handlungsbedarf und Anforderungsprofil 47
3.1 Ableitung des Handlungsbedarfs . . . 47
3.2 Zieldefinition . . . 49
3.3 Anwendungsfälle . . . 49
3.3.1 Integration des Verhaltens in den Digitalen Zwilling . . . 50
3.3.2 Steuerung des Digitalen Zwillings . . . 54
3.4 Definition des Anforderungsprofils . . . 57

4 Konzept der Virtuellen Klemmleiste 69
4.1 Konzeptübersicht . . . 69
4.2 Definition der Verhaltenslogik eines Digitalen Zwillings . . . 74
4.3 Virtualisierung der Verhaltenslogik . . . 80
4.3.1 Analyse der Verhaltenslogik . . . 81
4.3.2 Virtualisierung der Elektronik . . . 83
4.3.3 Definition der Virtuellen Klemmleiste . . . 87
4.4 Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik . 94
4.4.1 Auslegung der Architektur und Rahmenwerks des Digitalen Zwillings 95
4.4.2 Konfiguration von Partialmodellen . . . 98
4.4.3 Generierung ausführbarer Simulationen . . . 99
4.4.4 Verknüpfung zum Digitalen Zwillings-Master . . . 100
4.4.5 Verbindung mit Physischen Zwilling und Instanziierung . . . 101
4.5 Fazit . . . 102

5 Prototypische Implementierung 105
5.1 Virtualisierung der Verhaltenslogik . . . 107
5.1.1 Analyse der Verhaltenslogik . . . 107
5.1.2 Virtualisierung der Elektronik . . . 108
5.1.3 Virtuelle Klemmleiste . . . 112
5.2 Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik . 117
5.2.1 Architektur eines Digitalen Zwillings mit integrierter Verhaltenslogik117
5.2.2 Schnittstellen zwischen Partialmodellen und der Virtuellen Klemmleiste . . . 118

5.2.3 Verbindung mit dem Physischen Zwilling 123
5.3 Fazit 123

6 Validierung und Verifikation **125**
6.1 Anwendungsfall - Integration des Verhaltens in den Digitalen Zwilling . . . 125
6.2 Anwendungsfall - Steuerung des Digitalen Zwillings 128
6.3 Bewertung der Validierung 131
6.4 Verifikation des Anforderungsprofils 134
6.5 Fazit 136

7 Ausblick **141**

8 Zusammenfassung **143**

9 Anhang **145**
9.1 Anhang A - Implementierung OPC UA Client 145

Abbildungsverzeichnis

1.1 Symbolische Darstellung des Ziels der Dissertation mit dem Physischen Zwilling links und dem Digitalen Zwilling rechts 4

2.1 V-Modell für die Entwicklung Cyber-Physischer Systeme gemäß VDI/VDE 2206:2021 [156] 10
2.2 Struktur eines mechatronischen Systems gemäß VDI 2206 [158] 15
2.3 Erweiterung eines Assets zu einer I4.0-Komponente mittels einer Verwaltungsschale [45] 18
2.4 NASA Simulatoren als Abbildung der Raumfahrzeuge [Quelle: NASA] . . . 20
2.5 Verhaltenskategorien eines Systems nach Grieves und Vickers [59] 30
2.6 Soft- und Hardwareelemente eines typischen eingebetteten Systems [In Anlehnung an [13]] 43

3.1 Anwendungsfalldiagramm zur Integration der Verhaltenslogik in einen Digitalen Zwilling 51
3.2 Anwendungsfalldiagramm zur Steuerung des Digitalen Zwillings 55

4.1 Konzeptionellen Schritte bei der Integration der Verhaltenslogik in den Digitalen Zwilling mittels einer Virtuellen Klemmleiste 69
4.2 Konzeptübersicht 70
4.3 Aufteilung des Konzepts in zwei Teilprozesse 73
4.4 UML 2 Verhaltensdiagramme [in Anlehnung an [127]] 78
4.5 UML 2 Interaktionsdiagramme [in Anlehnung an [127]] 79
4.6 Teilprozess A1: Virtualisierung der Verhaltenslogik 80
4.7 Teilprozess A12: Virtualisierung der Elektronik 84
4.8 Teilprozess A13: Definition der Virtuellen Klemmleiste 87
4.9 Virtuelle Klemmleiste als Schnittstelle zur virtuellen Elektronik. An die Virtuellen Klemmen können weitere Partialmodelle des Digitalen Zwillings angebunden werden. 89
4.10 Modell der Virtuellen Klemmleiste 90
4.11 Modellierung einer Virtuellen Klemmleiste in XML 91

4.12 Ausschnitt aus einem OPC UA Informationsmodell mit integrierter Virtueller Klemmleiste im XML Format 93
4.13 Definition eines Datentyps für die Virtuelle Klemmleiste in OPC UA 94
4.14 Teilprozess A2: Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik 95
4.15 Übersicht der Modellelemente 96
4.16 Verknüpfung der Modelle des Digitalen Zwillings-Masters mittels der in OPC UA umgesetzten Virtuellen Klemmleiste 101

5.1 CNC Laserplotter als Demonstrator 105
5.2 Elektronische Komponenten des CNC Laserplotter. MEGA2560 Board (oben links), das RAMPS 1.4 (unten links) Board und die Pololu DRV8838 Motortreiber mit Kühlkörpern (rechts) im demontierten Zustand. 106
5.3 Modellierung- und Simulationswerkzeug für elektronische Komponenten *Proteus* von *Labcenter Electronics* 109
5.4 Konfiguration des Mikrocontrollers *ATMEGA2560* (links) und der Kommunikationsschnittstelle *COMPIM* in *Proteus* von *Labcenter Electronics* 111
5.5 Architektur des prototypischen Demonstrators 112
5.6 Ausschnitt des OPC UA Informationsmodells des CNC Laserplotters. 114
5.7 *UA Modeller* des Herstellers *Unified Automation GmbH* 115
5.8 Universeller OPC UA Client *UA Expert* des Herstellers *Unified Automation GmbH* 117
5.9 Betrachtete Elemente der Architektur der Virtuellen Klemmleiste 118
5.10 *NX - Mechatronic Concept Designer* von *Siemens* mit einem Modell des CNC Laserplotters 119
5.11 Signal Mapping des NX - Mechatronic Concept Designer von Siemens . . . 120
5.12 Ausschnitt der GPIOs des *MEGA 2560* Mikrocontroller Boards (links) und des *Monitor-Mikrocontroller* (rechts, ebenfalls ein *ATMEGA 2560*), welcher die Signale über das *MONITOR-COMPIM* mittels einer COM Verbindung weiterleitet. 121
5.13 Ausschnitt aus dem eingebetteten Code des Monitor-Mikrocontrollers. . . 122

6.1 Prozess der Integration des Verhaltens in den Digitalen Zwilling mit dem Fokus auf das Zusammenwirken der beteiligten Disziplinen. 126
6.2 Ausschnitt aus *cpu_map.h* des *Grbl-Mega* mit Darstellung der Zuordnung von Variablen zu entsprechenden Registern, PORTs und PINs 127
6.3 bCNC als Benutzungsoberfläche sowohl für das physische System als auch für den Digitalen Zwilling 129

6.4 Erweiterung der *bCNC* Software um einen OPC UA Client in der *__main__.py* 130
6.5 Validierung des Verhaltens des Digitalen Zwillings durch Vergleich mit dem physischen System 131
6.6 Sequenzdiagramm der Steuerung des Digitalen Zwillings durch den Nutzer 132

9.1 Implementierung der Funktion *remoteControl* zur Anbindung von *bCNC* an einen OPC UA Server in *control.py* (Fortsetzung auf der nächsten Seite) . . 145
9.2 Fortsetzung der Implementierung der Funktion *remoteControl* und die Implementierung der Funktion zur Trennung der Verbindung *quitOPCUA* in *__main__.py* von *bCNC* 146

Tabellenverzeichnis

3.1 Anforderungsprofil 66
3.2 Anforderungsprofil (Fortsetzung) 67

6.1 Verifikation des Anforderungsprofils 138
6.2 Verifikation des Anforderungsprofils (Fortsetzung) 139

Abkürzungsverzeichnis

3D	dreidimensional
AASX	Asset Administration Shell Explorer
ACOSAR	Advanced Co-Simulation Open System Architecture
AG	Aktiengesellschaft
API	Application Programmer Interface
ASIC	Application Specific Integrated Circuit
AutomationML	Automation Modeling Language
AUTOSAR	AUTomotive Open System ARchitecture
bzw.	beziehungsweise
CAD	Computer Aided Design
CASE	Computer Aided Software Engineering
CAx	Computer Aided x
CI	Continuous Integration
CNC	Computerized Numerical Control
COM Port	COMmunication Port
COMPIM	COMunication Port Physical Interface Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Of The Shelf

CPPS	Cyber-Physisches Produktions-System
CPS	Cyber-Physisches System
CT	Continuous Time
DCP	Distributed Co-Simulation Protocol
DE	Discrete Event
DIN	Deutsches Institut für Normung
DIS	Distributed Interactive Simulation
DT	Digital Twin
DTE	Digital Twin Environment
DTI	Digital Twin Instance
DTP	Digital Twin Prototype
DZ	Digitaler Zwilling
ECU	Electronic Control Unit
EDA	Electronic Design Automation
EN	Europäische Norm
FIT	Failure In Time
FMI	Functional Mockup Interface
FMU	Functional Mockup Unit
FPGA	Field Programmable Gate Array
G-Code	CNC Programmiersprache nach ISO 6983
GPIO	General Purpose Input-Output
GTV	Grieves' Test of Virtuality
HLA	High Level Architecture

HTTP	Hypertext Transfer Protocol
I4.0	Industrie 4.0
ID	Identifikator
IDL	Interface Definition Language
IKT	Informations- und Kommunikationstechnologien
IoT	Internet of Things
ISO	Intenational Organization for Standardization
ISS	Instruction Set Simulator
IEC	International Electrotechnical Commission
IP	Internet Protocol
ITEA2	Information Technology for European Advancement
JSON	JavaScript Object Notation
LCD	Liquid Crystal Display
MAP	Modelica Association Project
MCAL	Micro Controller Abstraction Layer
MISRA	Motor Industry Software Reliability Association
NASA	National Aeronautics and Space Administration
OMG	Object Management Group
OPC UA	Open Plattform Communicataions Unified Architecture
OPC UA CS	OPC UA Companion Specifications
ORB	Object Request Broker
OSGi	Open-Service-Gateway Initiative
OSI	Open Simulation Interface

OVP	Open Virtual Platforms
PDM	Produktdatenmanagement
PEP	Produktentstehungsprozess
PIN	Anschluss einer integrierten Schaltung
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
QoS	Quality of Service
RAM	Random Access Memory
RAMPS	RepRap Arduino MEGA Pololu Shield
REST	Representational State Tranfer
ROM	Read Only Memory
RTI	Run-Time-Infrastructure
RTOS	Real Time Operating System
RX	Recieving Exchange
SADT	Structured Analysis and Design Technique
SSP	System Structure and Parametrization
STEP	Standard for Exchange of Product Model Data
TCP	Transmission Control Protocol
TSN	Time-Sensitive Networking
TX	Transmission Exchange
UA	Unified Automation
UMATI	Universal Machine Technology Interface
UML	Unified Modeling Language

VDE	Virtual Development Environment
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
VDI	Verband Deutscher Ingenieure
vgl.	vergleiche
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WebGL	Web Graphics Library
WiGeP	Wissenschaftliche Gesellschaft für Produktentwicklung
XML	Extensible Markup Language
z. B.	zum Beispiel
ZIP	abgeleitet von *Zipper* - engl.: Reißverschluss

1 Einleitung

Die vorliegende Dissertation adressiert die Integration der Verhaltenslogik in den Digitalen Zwilling. Insbesondere mechatronische und Cyber-Physische Systeme besitzen durch die eingebettete Software und Elektronik eine komplexe Verhaltenslogik, die das Verhalten dieser Art von Systemen maßgeblich bestimmt. In Digitalen Zwillingen solcher Systeme wird diese Verhaltenslogik bisher entweder nicht berücksichtigt oder durch Ersatzmodelle angenähert (vgl. Kapitel 2.3). Im Falle der Nutzung von verhaltensbeschreibenden Ersatzmodellen führen veränderte Rahmenbedingungen zu einer Abweichung zwischen der tatsächlichen Verhaltenslogik und dem simulierten Verhalten, wodurch das virtuelle Abbild verfälscht wird. Anwendungsfälle des Digitalen Zwillings, die insbesondere das Verhalten des physischen Systems im Fokus haben, sind somit nicht möglich oder bedeutsam eingeschränkt.

In nachfolgenden Unterkapiteln sind die *(1.1) Motivation und Problemstellung* herausgearbeitet und die *(1.2) Zielsetzung* formuliert. Abschließend wird der *(1.3) Aufbau der Dissertation* erläutert.

1.1 Motivation und Problemstellung

Moderne und innovative mechatronische Produkte stellen die interdisziplinäre Produktentwicklung vor neue Herausforderungen. Die Disziplinen des Maschinenbaus, der Elektrotechnik/Elektronik und der Informatik arbeiten parallel oder sequenziell und teilen die jeweiligen Ergebnisse mit den anderen Disziplinen. Die mechatronischen Systeme kennzeichnen sich durch Wechselwirkungen der Mechanik, der Elektrik/Elektronik und der Informatik. Diese gegenseitige Beeinflussung tritt erst in der Gesamtheit des Systems in Erscheinung. Zusätzlich führt die Abhängigkeit zwischen Software und physischen Prozessen zu sogenannten Cyber-Physischen Systemen (CPS). Die Cyber-Physischen Systeme vereinen die physische Welt und die Datenverarbeitung und sind außerdem auf Basis von Kommunikationsschnittstellen mit anderen Systemen und dem Menschen vernetzt. Die Vernetzung bewirkt einen höheren Komplexitätsgrad, da nicht nur das System und der zugehörige physische Prozess für das Verhalten während des Betriebs relevant sind,

sondern auch die Interaktion mit externen Einflüssen. Diese Komplexität Cyber-Physischer Systeme lässt sich durch Modellbildung beherrscht.

Virtuelle Modelle werden jedoch nicht nur in der Entwicklungsphase eingesetzt, sondern finden auch in anschließenden Lebensphasen Anwendung. Durch die Kopplung von virtuellen Modellen mit einer Systeminstanz entsteht eine individuelle Abbildung, die als ein sogenannter Digitaler Zwilling zu bezeichnen ist. Zwischen einem Digitalen Zwilling und dem zugehörigen Physischen Zwilling besteht eine bidirektionale Verbindung, über die die beiden Zwillinge wechselwirken können.

Die einzelnen virtuellen Modelle unterschiedlichen Umfangs bilden jeweils Teile des physischen Systems ab. Die Vision des Digitalen Zwillings ist jedoch die ganzheitliche Abbildung. Im Falle von Cyber-Physischen Systemen bedeutet dies die Berücksichtigung der Wechselwirkungen der Modelle aus den Disziplinen der Mechanik, der Elektrik/Elektronik, der Informatik, des physischen Prozesses sowie der externen Kommunikationspartner[1].

Insbesondere die Abbildung des Verhaltens erfordert einen disziplinübergreifenden Ansatz. Das Verhalten eines Systems wird durch die eingebettete Elektronik, die eingebettete Software und die Mechanik des Systems mitbestimmt. Die eingebettete Software und die eingebettete Elektronik werden im Folgenden als Verhaltenslogik bezeichnet. So wird beispielsweise das Reaktionsverhalten eines Cyber-Physischen Systems auf eine Eingabe mittels einer Benutzungsoberfläche abhängig von der Verhaltenslogik und der Mechanik geformt. Wenn das Reaktionsverhalten eines Digitalen Zwillings auf die gleiche Eingabe möglichst identisch ausfallen soll, müssen demnach die Verhaltenslogik und die Mechanik im Digitalen Zwilling ebenfalls berücksichtigt werden. Durch die bidirektionale Verbindung könnte der Digitale Zwilling die eingebettete Software und die eingebettete Elektronik des physischen Systems verwenden und nur die Mechanik virtuell abbilden. Bei einer entkoppelten Nutzung des Digitalen Zwillings steht diese Option jedoch nicht zur Verfügung, sodass alle, das Verhalten beeinflussenden Teile, virtuell repräsentiert werden müssen. Während der Digitale Zwilling hinsichtlich der Mechanik bereits vielseitig erforscht ist, ist die vollständige virtuelle Abbildung des Verhaltens aufgrund der Wechselwirkung zwischen virtueller Verhaltenslogik sowie der virtuellen Mechanik bisher nicht durchdrungen. Doch gerade das originalgetreue Verhalten auf Basis der eingebetteten Software ermöglicht zukunftsweisende Anwendungsfälle des Digitalen Zwillings, wie beispielsweise die virtuelle Inbetriebnahme, virtuelle Absicherung von Anpassungen der eingebetteten Software oder das Testen von neuen Rahmenbedingungen und Eingaben.

Eine Hürde bei der Integration des Verhaltens eines eingebetteten Systems in dessen

[1] **Gender-Hinweis**: Zur besseren Lesbarkeit wird in dieser Hausarbeit das generische Maskulinum verwendet. Die in dieser Arbeit verwendeten Personenbezeichnungen beziehen sich – sofern nicht anders kenntlich gemacht – auf alle Geschlechter.

Digitalen Zwilling liegt darin begründet, dass die eingebettete Software dieser Systemart auf die entsprechende eingebettete Elektronik spezialisiert ist und sich nicht auf anderen Rechnerarchitekturen nativ ausgeführen lässt. Eingebettete Software, die für eine bestimmte Elektronikart entwickelt wurde, lässt sich demnach nicht direkt auf einem Rechner ausführen, auf dem andere Teile des Digitalen Zwillings betrieben werden. Hierfür muss stattdessen eine Simulation der entsprechenden eingebetteten Elektronik verwendet werden. Im Kontext des Digitalen Zwillings werden Elektroniksimulationen bisher jedoch nicht verwendet, sondern hauptsächlich in der Entwicklung von elektronischen Komponenten eingesetzt. Der Transfer dieser Technologie aus der Domäne der Elektronik und der Einsatz im Digitalen Zwilling bilden die Motivation der vorliegenden Dissertation.

Die Kernproblemstellung bei diesem Vorhaben ist die Konzeptionierung und Umsetzung der bisher fehlenden Schnittstelle zwischen einer Elektroniksimulation und den darin anknüpfenden Teilen eines Digitalen Zwillings, die im Folgenden als eine virtuelle Software-Hardware Schnittstelle bezeichnet wird. Zur Lösung der disziplinübergreifenden Problemstellung sind außerdem sowohl die Informatik, als auch die Elektrotechnik und der Maschinenbau hinzuzuziehen, da jede Domäne das entsprechende Fachwissen bereitstellen muss. Zur Koordination der Zusammenarbeit wird demnach neben der Entwicklung einer virtuellen Software-Hardware Schnittstelle auch eine übergreifende Vorgehensweise in Form einer Prozessbeschreibung benötigt.

1.2 Zielsetzung

Die vorliegende Dissertation hat die Integration der Verhaltenslogik eines physischen Systems in dessen Digitalen Zwilling zum Ziel. Handlungsbedarf besteht insbesondere auf dem Gebiet mechatronischer und Cyber-Physischer Systeme, da diese durch ein hohes Maß an Komplexität der Verhaltenslogik gekennzeichnet sind. Diese Verhaltenslogik, bestehend aus der eingebetteten Software und der eingebetteten Elektronik des Systems, soll nun auch in den Digitalen Zwilling integriert werden, um ein, dem physischen System möglichst identisches, dynamisches Verhalten zu ermöglichen. Um das angestrebte Ziel zu erreichen, sind wissenschaftliche Grundlagen zu beschreiben, eine domänenübergreifende Schnittstelle und ein Vorgehen beschrieben werden, welche zu diesem Ziel führen.

In Abbildung 1.1 ist das Ziel symbolisch dargestellt. Gemäß des Zielbildes soll ein Digitaler Zwilling mit einer virtuellen Elektronik ausgestattet werden, welche die gleiche eingebettete Software integrieren und einsetzen kann, wie der Physische Zwilling. In dieser Vision soll ein Nutzer das Verhalten eines Digitalen Zwillings mittels einer Benutzungsoberfläche auf die gleiche Weise beeinflussen können, wie die Steuerung des physischen Systems. Dies setzt voraus, dass die eingebettete Software sich in den Digitalen

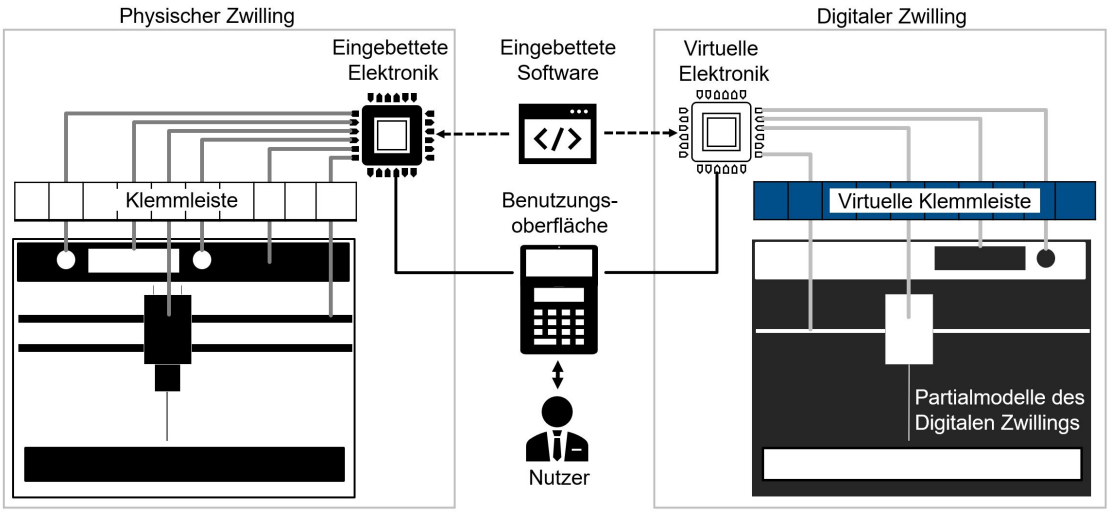

Abbildung 1.1: Symbolische Darstellung des Ziels der Dissertation mit dem Physischen Zwilling links und dem Digitalen Zwilling rechts

Zwilling integrieren lässt. Die Integration erfordert wiederum eine virtuelle Repräsentation der eingebetteten Elektronik, welche eine geeignete Umgebung zur Installation der eingebetteten Software bietet. Gemäß der Abbildung 1.1 ist die virtuelle Elektronik mit unterschiedlichen Partialmodellen des Digitalen Zwillings verknüpft und ermöglicht letztlich die Steuerung. Diese Verknüpfung erfolgt mittels einer Software-Hardware Schnittstelle, die im Rahmen der Dissertation unter der Bezeichnung *Virtuelle Klemmleiste* entwickelt wird.

Zur Erreichung des Ziels sind folgende Fragestellungen zu untersuchen:

- Wie lässt sich die eingebettete Software des physischen Systems in dessen Digitalen Zwilling integrieren?
- Wie muss dabei die eingebettete Elektronik des physischen Systems berücksichtigt werden?
- Wie ist die Schnittstelle zwischen der virtuellen Verhaltenslogik und den damit verknüpften Partialmodellen des Digitalen Zwillings auszubilden?
- Welche Disziplinen müssen bei dieser interdisziplinären Fragestellung mitwirken?
- Wie ist die disziplinübergreifende Vorgehensweise zur Integration der Verhaltenslogik?

- Welche IT-Werkzeuge werden benötigt, um das Ziel zu erreichen?

Die Beantwortung dieser Fragestellungen zur Erreichung des Ziels und die Lösung der damit verbundenen Aufgaben führt zum Inhalt der Dissertation. Der Kern der Dissertation besteht in der Erarbeitung der notwendigen Schnittstelle und der Beschreibung einer Vorgehensweise zur Integration der Verhaltenslogik in den Digitalen Zwilling.

1.3 Aufbau der Dissertation

Zur Beantwortung der formulierten wissenschaftlichen Fragestellungen gliedert sich die vorliegende Dissertation in acht Kapitel.

In Kapitel 1 sind die Motivation und die Problemstellung beschrieben, die in einem Forschungsbedarf resultieren. Um die wissenschaftlichen Fragestellungen zu beantworten, wird ein konkretes Ziel formuliert.

In Kapitel 2 erfolgt die Beschreibung wissenschaftlicher Grundlagen und des Stands der Technik und Forschung auf dem, im Rahmen dieser Dissertation behandelten, Wissensgebieten. Zu diesen gehören insbesondere die Virtuelle Produktentwicklung und Themen der Digitalisierung mit dem Schwerpunkt auf dem Digitalen Zwilling. Weiterhin werden, die in dem Kontext relevanten, Grundlagen zum Thema Simulation ausgearbeitet.

In Kapitel 3 ist der Handlungsbedarf weiter konkretisiert. Unter Berücksichtigung von Anwendungsfällen werden Anforderungen an die Konzeption und die Implementierung ausformuliert, welche in einem Anforderungsprofil münden.

In Kapitel 4 wird, unter Berücksichtigung des Anforderungsprofils, das Konzept der *Virtuellen Klemmleiste* als ein Ansatz zur Integration der Verhaltenslogik in den Digitalen Zwilling entwickelt. Die Konzeption besteht aus der Prozessbeschreibung, der Vorgehensweise bei der Integration und der Spezifikation einer *Virtuellen Klemmleiste*.

In Kapitel 5 wird, zur Überprüfung der Tragfähigkeit des entwickelten Konzepts, eine prototypische Implementierung vorgestellt. Die Implementierung ist anhand eines exemplarischen Demonstrators in Form eines numerisch gesteuerten Laserplotters realisiert.

In Kapitel 6 erfolgt die Validierung der Tragfähigkeit des Konzepts anhand von Anwendungsfällen, die auf Basis der Implementierung durchgeführt werden.

In Kapitel 7 wird ein Ausblick auf offene Potenziale und Möglichkeiten der Erweiterung des vorgestellten Konzepts gegeben und die daraus resultierenden Forschungsrichtungen aufgezeigt.

In Kapitel 8 erfolgt eine Zusammenfassung der Dissertation.

2 Stand der Technik und Forschung

Die Entwicklung des Konzepts einer Virtuellen Klemmleiste als einen Ansatz zur Integration der Verhaltenslogik in den Digitalen Zwilling basiert auf Erkenntnissen aus unterschiedlichen Disziplinen, an deren Schnittstellen das Konzept zu verorten ist. Die Dissertation konzentriert sich speziell auf Cyber-Physische Systeme mit dem Fokus auf die Mechanik, Elektrotechnik und Software. Dieses Kapitel stellt den Stand der Technik und Forschung, der für das Verständnis des darauffolgenden Inhalts notwendig ist. Den Ausgangspunkt bildet die Virtuelle Produktentwicklung mit dem Fokus auf den Produktentwicklungsprozess und speziell die Entwicklung eingebetteter Software. Diese Grundlagen sind notwendig, um Einblicke in die einzelnen Disziplinen zu erhalten. Anschließend erfolgt eine Darstellung von Themen der Digitalisierung und Industrie 4.0. Darin findet eine Konkretisierung des Cyber-Physischen Systems sowie der Verwaltungsschale statt. Im Anschluss erfolgt die Schärfung des Konzepts durch Definitionen und das Aufzeigen der wesentlichen Charakteristika des Digitalen Zwillings. Des Weiteren sind Ansätze zum Design sowie speziell die Verhaltensabbildung im Digitalen Zwilling beschrieben und die Bereitstellung Digitaler Zwillinge eingeführt. Aufbauend auf diesen Grundlagen wird im nächsten Kapitel die Simulation im Digitalen Zwilling tiefer behandelt, indem die Bedeutung von Co-Simulation ausgearbeitet und die geeigneten Architektur- und Schnittstellenstandards zusammengefasst werden. Als Technologien im Digitalen Zwilling werden abschließend FMI (*Functional Mockup Interface*), OPC UA (*Open Platform Communications Unified Architecture*) sowie Virtuelle Plattformen aus der Disziplin der Entwicklung eingebetteter Hard- und Software erörtert.

2.1 Virtuelle Produktentwicklung

Die in der vorliegenden Dissertation betrachteten Systeme sind technische Produkte. Ein Produkt ist gemäß der Definition nach VDI 4520 [155] ein:

„Gegenstand und/oder Dienstleistung, den/die ein Unternehmen aktuell und zukünftig anbietet und der/die für Kunden oder potenzielle Kunden einen Wert besitzt“ [155].

Im Folgenden liegt der Fokus auf technischen Produkten in Form von technischen Systemen, welche mechanische, elektronische und Software-Komponenten aufweisen. Die einzelnen Elemente bilden einen hierarchischen Zusammenschluss zu übergeordneten Produktstrukturen [20].

Technische Produkte entstehen in einem Produktentstehungsprozess, der sich nach Anderl [8] in die Phasen der *Produktplanung*, *Entwicklung und Konstruktion*, *Arbeitsvorbereitung* und *Produktherstellung* unterteilt. Zusammen mit den anschließenden Phasen des *Produktvertriebs*, der *Produktnutzung* und *Produktrecycling und -entsorgung* resultiert ein Produktlebenszyklus, in dem technische Produkte iterativ weiterentwickelt werden.

In jeder Phase des Produktlebenszyklus entstehen Daten, die mit den Prinzipien und Methoden der Produktdatentechnologie ausgetauscht, gespeichert, archiviert und transformiert werden. Die Produktdatentechnologie basiert auf dem Produktdatenmodell, welches in der ISO-Normenreihe 10303 definiert ist [69]. Das Produktdatenmodell setzt sich aus der Produktdefinition, Produktrepräsentation und Produktpräsentation zusammen [12]. Die Produktdefinition umfasst die administrativen und organisatorischen Daten. Die Produktrepräsentation enthält die Produktdaten zur rechnerverarbeitbaren Abbildung von Produktmerkmalen. Die Daten zur grafischen oder textuellen Darstellung der Produktrepräsentation werden unter Produktpräsentation zusammengefasst. Die durchgängige Nutzung der Produktdaten über alle Produktlebensphasen hinweg manifestiert sich nach Anderl in einem *integrierten Produktdatenmodell* [7].

Die Produktdaten werden im Rahmen des *Produktdatenmanagement* (PDM) in PDM Systemen verwaltet. Mit dem besonderen Augenmerk auf die lebensphasenübergreifende Verknüpfung und Verwendung der Produktdaten, werden diese Systeme zu *Product Lifecycle Management* (PLM) Systemen ausgeweitet [80]. Ein PLM System stellt dadurch die Wissensbasis eines Unternehmens bezüglich Produktdaten und Prozessen über alle Produktlebensphasen - von der Produktidee bis Recycling - dar [19]. In einem interdisziplinären Prozess, wie der Entwicklung von Cyber-Physischen Systemen, können somit die beteiligten Disziplinen, wie die Domänen der Mechanik, der Elektrik und Elektronik, die Softwareentwicklung sowie weitere, kontextspezifische Fachrichtungen, auf eine gemeinsame Datenbasis im PLM zugreifen [156].

2.1.1 Produktentwicklungsprozess

Die Phasen des Produktlebenszyklus von der Produktplanung bis zur Produktherstellung werden im *Produktentstehungsprozess* (PEP) zusammengefasst. Der PEP ist eine Prozesskette, deren einzelnen Schritte definierte Funktionen aufweisen und die mittels Schnittstellen zu benachbarten Prozessschritten verbunden ist [19]. Grundsätzlich existieren unter-

schiedliche Methoden der Produktentwicklung, die sich je nach Domäne unterscheiden. In der Mechanik sind beispielsweise andere Methoden verbreitet, als in der Softwareentwicklung, wobei teilweise die Vorgänge aus der jeweils anderen Domäne mit Anpassungen übernommen werden. Das, in der VDI 2221 beschriebene, generelle Vorgehen beim Entwickeln und Konstruieren wurde beispielsweise in der Softwareentwicklung aufgegriffen und fand später Eingang in das V-Modell der Entwicklung mechatronischer Produkte nach der VDI 2206 [158]. Der durch den *Verband Deutscher Ingenieure* (VDI) erarbeitete Leitfaden wurde im Jahr 2021 schließlich zusammen mit dem *Verband der Elektrotechnik, Elektronik und Informationstechnik* (VDE) überarbeitet und berücksichtigt nun auch die Entwicklung Cyber-Physischer Systeme [156]. Damit adressiert der überarbeitete Leitfaden genau die Art Systeme, die in der vorliegenden Dissertation bedeutsam sind und wird im Folgenden in den Grundzügen erläutert. Der PEP bildet den Vorgang, während dessen die Modelle und Daten entstehen, die anschließend im Digitalen Zwilling integriert und instanziiert werden.

Das V-Modell nach VDI/VDE 2206 verbindet die klassischen Disziplinen der Mechanik, Elektrotechnik und Softwaretechnik in einer gemeinsamen Entwicklungsmethodik und hat zum Ziel, die jeweils individuellen Terminologien, Methoden und Fachwissen in einem interdisziplinären Prozess zu bündeln [156]. Indes besteht die Besonderheit Cyber-Physischer Systeme in der Flexibilität und Wandlungsfähigkeit aufgrund der Vernetzung im Betrieb. Dies muss in besonderem Maße bereits während der Entwicklung berücksichtigt und hinsichtlich der Zuverlässigkeit und Sicherheit abgesichert werden [156]. Durch die große Anzahl der Systemelemente und dessen Verbindungen steigt die Komplexität [57] und erfordert eine effektive virtuelle Absicherung. Das V-Modell ist in der Abbildung 2.1 dargestellt.

Das V-Modell macht keine zeitlichen Vorgaben und gibt auch keine sequenzielle Reihenfolge vor, sondern beschreibt die Abhängigkeiten der Entwicklungsaufgaben [156]. Es besteht aus drei parallelen Strängen: dem Strang des Anforderungsmanagement, der Modellierung und Analyse und dem mittleren Strang als die Kernaufgabe der Systementwicklung. Zusätzlich sind im V-Modell sechs Kontrollpunkte integriert, die mittels Kontrollfragen Rückschlüsse auf den Reifegrad des Produkts schließen lassen. In jedem der Stränge findet eine Zusammenarbeit zwischen den einzelnen Disziplinen statt, sodass Abhängigkeiten bestehen, die berücksichtigt werden müssen [156]. Während der gesamten Entwicklung findet kontinuierlich eine Verifikation und Validierung des Systems statt, für die insbesondere die virtuellen Modelle herangezogen werden. Dabei ist die Berücksichtigung der Virtuellen Produktentwicklung als Teil des V-Modells, die durch Anderl et al. [10] in Form des W-Modells erarbeitet wurde, bereits in das aktualisierte V-Modell integriert. Die Virtuelle Produktentwicklung ist somit ein wesentlicher Teil des V-Modells und dient nicht nur der Entwicklung des Produkts im engeren Sinne, sondern umfasst

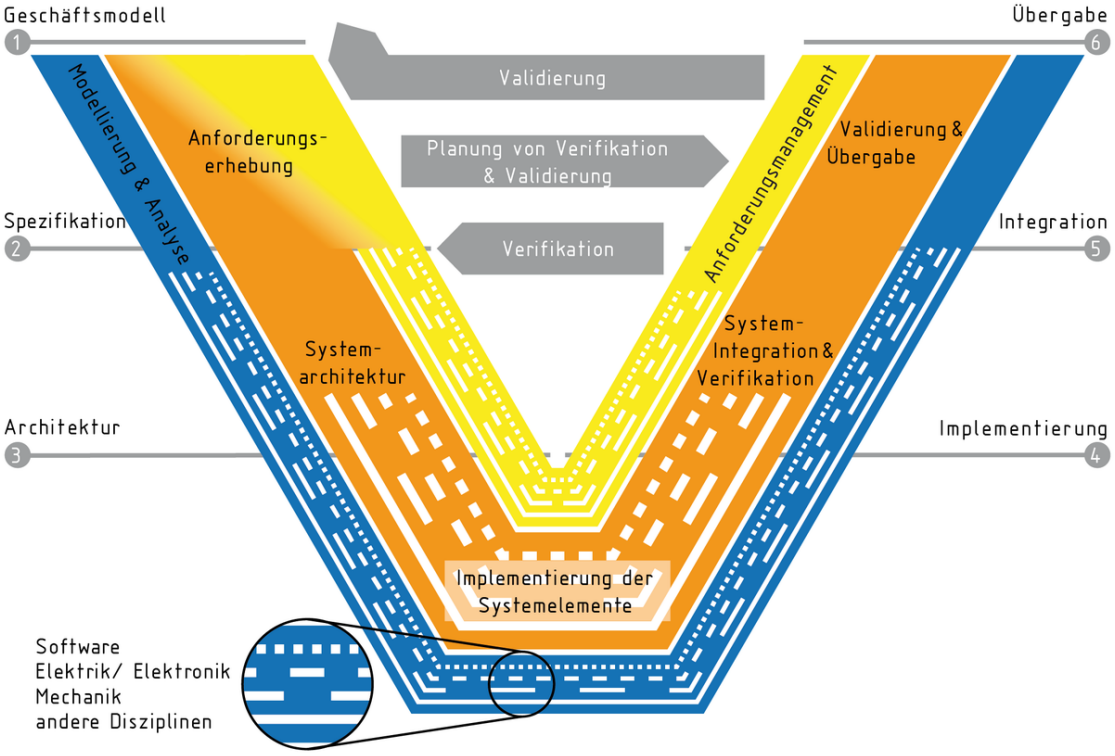

Abbildung 2.1: V-Modell für die Entwicklung Cyber-Physischer Systeme gemäß VDI/VDE 2206:2021 [156]

auch die Dienste, die im Rahmen von Produkt-Service Systemen entwickelt werden. Zu diesen Diensten gehört auch der Digitale Zwilling, dessen Grundlage die Modelle und Daten aus dem Entwicklungsprozess sind [143].

2.1.2 Entwicklung eingebetteter Software

Die Softwaretechnik ist Teil der Entwicklung eines Cyber-Physischen Systems und implementiert einige Besonderheiten dieser Systemart. Sowohl die Steuerungssoftware des Systems, als auch die Kommunikationsnetzwerke und Services sind softwarebasiert. Daher erfolgt zusätzlich eine Erläuterung der Softwareentwicklung als Disziplin, da diese neuartige Methoden, wie beispielsweise die Agile Softwareentwicklung, einsetzt, die in der Entwicklungsmethodik nach VDI/VDE 2206 nicht gesondert betrachtet werden aber für die vorliegende Dissertation bedeutsam sind. Software besitzt einen Softwarelebens-

zyklus, welcher in Analogie zum Produktlebenszyklus in die Phasen: *Planung*, *Definition*, *Entwurf*, *Implementierung*, *Abnahme / Einführung*, *Anwendung*, *Pflege und Wartung* und *Migration / Stilllegung* zu unterteilen ist [6]. Je nach Sichtweise wird der Software-Entwicklungsprozess auch in die vier Kernphasen *Analyse*, *Entwurf*, *Implementierung* und *Test* eingeteilt. Diese Phasen lassen sich weiterhin ergänzen um die *Integration*, *Konsolidierung*, *Dokumentation* sowie *Betrieb und Wartung* [53]. Aufgrund der Durchgängigkeit und Analogie zum Produktlebenszyklus wird im Folgenden jedoch der Softwarelebenszyklus nach Anderl [6] zugrunde gelegt. Die komplexe Struktur der Produkte und der Prozesse der Softwareentwicklung erfordern Vorgehensmodelle, um erfolgreich Software entwickeln zu können. Nach Gessler können die Vorgehensmodelle in klassische und moderne Modelle unterschieden werden [53]. Zu den klassischen Modellen gehören das *Wasserfall-* und das *V-Modell* [70] der Softwareentwicklung. Unter modernen Modellen sind das *evolutionäre Modell*, das *Prototypen-Modell*, das *Komponenten-Modell*, die *inkrementellen Modelle* und das *Spiral-Modell* zu nennen [53]. Eine ausführliche Beschreibung der Vorgehensmodelle kann weiterführender Literatur [140, 130] entnommen werden. In der Klasse der inkrementellen Modelle sind auch die iterativen, agilen Entwicklungsmethoden zu verorten. Innerhalb der agilen Modelle sind das *Extreme Programming*, *Kanban* und *Scrum* bekannte Vertreter. Insbesondere hat die Scrum Methode in den letzten Jahren einen hohen Verbreitungsgrad erreicht [142]. Bei agiler Softwareentwicklung im Kontext Cyber-Physischer Systeme müssen die Besonderheiten iterativer und inkrementeller Softwareentwicklung berücksichtigt werden, da es maßgebliche Auswirkungen auf die Zusammenarbeit im interdisziplinären Prozess hat. Durch die kontinuierliche Weiterentwicklung der Software, die auch als *Continuous Integration* (CI) [21] bekannt ist, und durch mögliche Anpassungen und Veränderungen der Software während des Betriebs in Form von Updates, ist auch das gesamte System im Betrieb Veränderungen unterworfen. Dies muss in allen Produktlebensphasen des Produkts berücksichtigt werden [120, 159].

Die Entwicklung von Software für eingebettete Systeme unterliegt besonderen Herausforderungen. Einige dieser Besonderheiten sind die Kopplung der Datenverarbeitung an die physikalischen Prozesse, die Qualitäts- und Zuverlässigkeitsanforderungen, der Zusammenschluss zu verteilten Systemen und oft das Fehlen eines Betriebssystems [53]. Bei der Entwicklung eingebetteter Software müssen die Rahmenbedingungen des physikalischen Prozesses, mit dem das eingebettete System interagieren wird, bekannt sein. In Abhängigkeit vom Einsatz unterliegt die Datenverarbeitung zeitlichen Anforderungen, die unter Echtzeit zusammengefasst werden. Cyber-Physische Systeme sind demnach Echtzeitsysteme und setzen sich aus einem Prozess und einem Datenverarbeitungsgerät zusammen. Gerade die Wechselwirkung zwischen einem kontinuierlichen Prozess und der diskreten Datenverarbeitung stellt eine besondere Herausforderung bei der Entwicklung Cyber-PhysischerSysteme dar [90]. Um dies zu berücksichtigen geht bei Cyber-Physischen

Systemen, im Gegensatz zu den meisten Softwaresystemen, die Zeit als Eingabe in die Verarbeitung mit ein. Dadurch ist zu gewährleisten, dass die Datenverarbeitung in festgelegten Zeitschranken durchführbar ist. Das erfordert den Einsatz von deterministischen Systeme. Hierbei wird zwischen weicher und harter Echtzeit unterschieden. Während bei harter Echtzeit die Gefährdung für Menschen und Dinge infolge von zeitlichem Fehlverhalten ausgeschlossen werden muss, lässt sich bei weicher Echtzeit Fehlverhalten tolerieren und als Qualitätsmerkmal ansehen [139, 53].

Neben der zeitlich korrekten Funktionsweise müssen allgemein die Qualitätsmerkmale sichergestellt werden. Ein in der vorliegenden Dissertation wesentliches Qualitätsmerkmal ist die Zuverlässigkeit. Die Zuverlässigkeit betrifft das gesamte System und setzt sich aus den Zuverlässigkeiten der Einzelteile zusammen. Die Software ist in den meisten Fällen komplexer als die elektronischen Teile und verursacht bis zu mehrere Größenordnungen mehr Fehler und Ausfälle, als die verwendete Hardware [55]. Demnach wird der Verifikation und dem Testen von Software mehr Bedeutung beigemessen, da es den größten Anteil bei der Sicherstellung der Zuverlässigkeit einnimmt. Die Zuverlässigkeit wird als die Wahrscheinlichkeit definiert, das ein System seine Funktion unter erwarteten Randbedingungen erfüllt und es nicht zu Systemausfällen kommt [139]. Dabei wird zwischen der fehlerhaften Handlung (engl. *error*), dem inneren Fehler (engl. *fault*) und der Fehlerwirkung bzw. dem äußeren Fehler (engl. *failure*) unterschieden. Die Rate der äußeren Fehler wird in *Failure In Time* (FIT) gemessen, welche die Rate der auftretenden Fehler pro 10^9 Betriebsstunden angibt [139].

Das Produkt der Softwareentwicklung ist Code für ein System. Dieser liegt zunächst in einer menschenlesbaren Form vor, welche als Quellcode bezeichnet wird. Die Programme für Mikrocontroller werden häufig in der Programmiersprache *C* entwickelt [53]. Vor dem Hochladen des Codes auf den Mikrocontroller wird der Quellcode jedoch in einem Übersetzungsprozess in Maschinencode transformiert. Die wesentlichen Aufgaben übernehmen der sogenannte *Compiler*, *Assembler* und *Linker/Locator*, die das Quellprogramm in ein Zielprogramm in Maschinencode überführen [53]. Bei der Softwareentwicklung werden sogenannte *Computer Aided Software Engineering* (CASE) Werkzeuge eingesetzt. Diese unterstützen die unterschiedlichen Aufgaben in den Phasen des Software-Engineerings, wie die Analyse, Entwurf, Implementierung und Test [53]. Im Rahmen der Dissertation wird der Entwicklungsprozess eingebetteter Software nicht näher betrachtet, sondern vom fertigen Softwareprodukt ausgegangen. Bei eingebetteten Systemen ist ein Betriebssystem nicht zwangsweise notwendig, da diese auf eine Aufgabe spezialisiert sind, die möglichst ressourceneffizient ausgeführt werden sollen. Bei fehlendem Betriebssystem übernimmt die Applikation selbst die notwendigen Aufgaben wie beispielsweise Speicherverwaltung und ist dadurch auf die verwendete Hardwarearchitektur spezialisiert [53]. Dies bringt weitere Herausforderung für die Entwicklung eingebetteter Systeme, da der Entwickler

nicht von der abstrahierenden Funktion eines Betriebssystems und Treiber profitieren kann, sondern die Hardware kennen muss.

2.1.3 Entwicklung der Elektrotechnik und Elektronik

Die Software von Cyber-Physischen Systemen, welche eine im Rahmen dieser Dissertation betrachtete Systemart darstellen, wird auf eingebetteter Elektronik ausgeführt. Die Elektronik, welche auch als Hardware bezeichnet wird, besteht im Wesentlichen aus einem Prozessor, einem Speicher und aus Peripheriebausteinen [6]. Zwei für Cyber-Physische Systeme wichtige Rechnergruppen stellen die Einplatinencomputer, welche die notwendigen Hardwarekomponenten auf einer Platine vereinen, und Prozessrechner, die auf die Verarbeitung analoger Signale und die Steuerung und Regelung von Motoren und Stellgliedern spezialisiert sind, dar [6]. Bei der Entwicklung eingebetteter und Cyber-Physischer Systeme lässt sich entweder auf bereits existierende Hardware, die als *Comercial Of The Shelf* (COTS) bezeichnet wird, ganz oder teilweise zugreifen oder die elektronischen Komponenten weitgehend eigenständig - und auf einen konkreten Anwendungsfall spezialisiert - entwickeln. Eine aus elementaren elektronischen Bausteinen entwickelte Elektronik wird als Application Specific Integrated Circuit (ASIC) bezeichnet [53]. Eine weitere Möglichkeit stellt die Verwendung von Field Programmable Gate Arrays (FPGAs) dar. Hierbei werden zunächst unspezifische Logikschaltkreise durch softwarebasierte Konfiguration auf einen Einsatzzweck ausgelegt. Die Software konfiguriert die Verdrahtung der Hardware [53]. Die dabei verwendete Software unterscheidet sich wesentlich von den Programmiersprachen von Mikroprozessoren. Die *Verilog Hardware Description Language* (kurz *Verilog*) und die ***V**ery High Speed Integrated Circuit (VHSIC) **H**ardware **D**escription **L**anguage* (*VHDL*) sind Beschreibungssprachen zur Konfiguration von Schaltungen eines FPGAs [53]. Durch den grundlegenden Unterschied von FPGAs gegenüber von Mikrocontrollern, werden diese im Rahmen der Dissertation nicht weiter betrachtet.

Die wesentlichen Aufgaben bei der Entwicklung eingebetteter Systeme bestehen aus der *Analyse und Verständnis der Entwicklungsaufgabe*, der *Identifikation relevanter Randbedingungen* und der *Erarbeitung einer Lösung unter den gegebenen Randbedingungen* [53]. Die Randbedingungen einer Entwicklungsaufgabe hängen vom konkreten Produkt ab. Es können beispielsweise *technische*, *system-technische*, *ökonomische* und *diverse* Randbedingungen unterschieden werden [53]. Während des Entwicklungsprozesses muss ein Kompromiss zwischen konträren Anforderungen eingegangen werden, um eine optimale Lösung zu finden. So besteht beispielsweise ein Zielkonflikt zwischen Datenrate und Ressourcenverbrauch, da mit einer steigenden Datenrate auch ein erhöhter Energieverbrauch einhergeht. Die konkrete Vorgehensweise bei der Entwicklung von FPGA und ASIC ist ein komplexer und sich verändernder Prozess. Für detailliertes Studium wird auf

weiterführende Literatur verwiesen [53].

Schöfer [131] entwickelte ein Konzept zur Absicherung von Steuergerätesoftware in mechatronischen Produkten durch Verknüpfung des Software-Modells mit Geometrie-Modellen in einer gekoppelten Model-in-the-Loop Simulation. Mit dem entwickelten Vorgehen kann, insbesondere in frühen Phasen der Produktentwicklung, die Korrektheit der Reaktion der Software auf geänderte Randbedingungen überprüft werden [131]. Das Konzept bildet nicht die Zeit und damit auch nicht die Echtzeitbedingungen ab. Damit kann das vollständige dynamische mechatronische Systemverhalten nicht untersucht werden.

2.2 Digitalisierung und Industrie 4.0

Die steigende Durchdringung des industriellen Umfeldes durch Konzepte der Digitalisierung und Vernetzung löste einen Wandel der Geschäftsmodelle und Dienstleistungen aus, welcher unter der Bezeichnung Industrie 4.0 zusammengefasst wird [121]. Sowohl die Produktionssysteme sind auf unterschiedlichen Ebenen, vom intelligenten Sensor [22], über die intelligente Fabrik [1], bis zu intelligenten Wertschöpfungsnetzwerken [122, 48], miteinander vernetzt, als auch die Produkte selbst [11]. Durch die globale Vernetzung der Systeme und die enge Verknüpfung zwischen Datenverarbeitung und physikalischen Prozessen müssen Produktionssysteme neuen Anforderungen gerecht werden [90, 34]. Die Weiterentwicklung der Systeme resultiert in einer neuen Systemart, welche als Cyber-Physisches System bezeichnet wird. Die fortschreitende Digitalisierung in der Produktentwicklung ermöglicht die Generierung von virtuellen Modellen von Systemen und Produkten, die sich über die Entwicklungsphase hinaus für neue Geschäftsmodelle verwenden lassen. Dieser Trend führte zu der Vision eines digitalen Abbildes eines Systems oder eines Produkts - dem Digitalen Zwilling. In den folgenden Kapiteln werden die Grundlagen des Cyber-Physischen Systems und des Digitalen Zwillings eingehender erläutert, da diese für das im Rahmen dieser Dissertation entwickelte Konzept von Bedeutung sind.

2.2.1 Cyber-Physisches System

Cyber-Physische Systeme sind die Weiterentwicklung von mechatronischen Systemen [39]. Sie ergänzen den grundlegenden Aufbau von mechatronischen Systemen durch zusätzliche Funktionen. Die Struktur eines mechatronischen Systems wird in der VDI Richtlinie 2206 *Entwicklungsmethodik für mechatronische Systeme* [158] beschrieben. Ein mechatronisches System besteht aus vier Elementen: dem Grundsystem, den Sensoren, den Aktoren und der Informationsverarbeitung (siehe Abbildung 2.2). Ein Aktor wirkt

mittels eines Energieflusses auf das physische Grundsystem ein und erreicht dadurch einen Zustandswechsel. Der Zustandswechsel wiederum wird durch Sensoren wahrgenommen und an die Informationsverarbeitung in Form eines Informationsflusses weitergeleitet. Auf Basis dessen steuert die informationsverarbeitende Einheit die Aktoren. Ein Kommunikationssystem zu externen informationsverarbeitenden Einheiten sowie eine Mensch-Maschine Schnittstelle sind bei einem mechatronischen System optional.

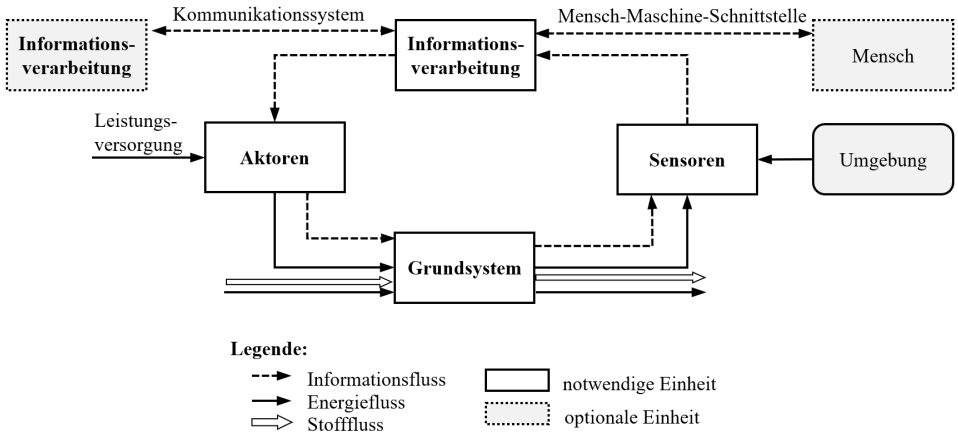

Abbildung 2.2: Struktur eines mechatronischen Systems gemäß VDI 2206 [158]

Mit dem stetigen Zusammenwachsen der Informations- und Kommunikationstechnologien (IKT) mit den Technologien aus der Produktion [25], der sogenannten *Operational Technology*, nimmt die externe Kommunikation sowie die Interaktion zwischen den physikalischen Prozessen und der Informationsverarbeitung eine immer bedeutsamere Rolle ein. Physikalische Prozesse lassen sich über das Internet analysieren und beeinflussen [33]. Es entsteht eine enge Verbindung digitaler Systeme und Modelle mit Objekten und Prozessen der realen Welt. Der Name *Cyber-Physisches System* basiert auf den Begriffen *cyberspace*, welcher seinen Ursprung in der Science-Fiction hat [118] und *cybernetics* [91], welcher die Steuerung und Regelung von Maschinen beschreibt. Die optionalen Elemente Kommunikationssystem und Mensch-Maschine-Schnittstelle eines mechatronischen Systems sind für ein Cyber-Physisches System somit obligatorisch. Der Unterschied liegt jedoch nicht nur in der Optionalität, sondern auch in der Art der Systemelemente. Während die Kommunikation eines mechatronischen Systems grundsätzlich einen geringen Umfang, wie beispielsweise die lokale Verbindung mit einem übergeordnetem Analysesystem, aufweisen kann, ist die Kommunikation eines Cyber-Physischen System

weitreichender. Diese hat als ein besonderes Merkmal der Cyber-Physischen Systeme die globale, internet-basierte Vernetzung zum Ziel [5]. Die Vernetzung mittels digitaler Netzwerke und die damit verbundene weltweite Nutzung von Daten und Diensten sowie die multimodale Mensch-Maschine Schnittstelle bilden die Grundlage für die zukünftige industrielle Entwicklung in Form von Industrie 4.0 [75]. Die überarbeitete VDI/VDE 2206 stellt ein Cyber-Physisches System in einer Definition wie folgt dar [156]:

„Cyber-Physische Systeme sind Systeme mit eingebetteter Software (als Teil von Geräten, Gebäuden, Verkehrsmitteln, Verkehrswegen, Produktionsanlagen, medizinischen Prozessen, Logistik-, Koordinations- und Managementprozessen), die

- *über Sensoren unmittelbar physikalische Daten erfassen und durch Aktoren auf physikalische Vorgänge einwirken,*
- *erfasste Daten auswerten und speichern und aktiv oder reaktiv mit der physikalischen sowie der digitalen Welt interagieren,*
- *über digitale Kommunikationseinrichtungen untereinander sowie in globalen Netzen verbunden sind (drahtlos und/oder drahtgebunden, lokal und/oder global),*
- *weltweit verfügbare Daten und Dienste nutzen und*
- *über eine Reihe dezidierter, multimodaler Mensch-Maschine-Schnittstellen verfügen.“*

Geisberger und Broy akkumulieren in der *acatech Studie* [52] die Anwendungsgebiete, Fähigkeiten und Charakteristika von Cyber-Physischen Systemen und beschreiben die Forschungsthemen sowie deren Bedeutung für den Standort Deutschland. Dabei wird die Entwicklung hin zu zunehmender Komplexität und Autonomie beschrieben [52]. An die Cyber-Physischen Systeme werden Anforderungen und Erwartungen gestellt. Monostori fasst einige dieser Erwartungen, wie beispielsweise Robustheit, Selbstorganisation, Sicherheit, Steuerung in Echtzeit, Transparenz und Vorhersagbarkeit, Effizienz und Korrektheit der Modelle, zusammen [108]. Monostori et al. arbeiten diese Anforderungen noch weiter aus und stellen diese in den Kontext von Industrie 4.0 [109]. Broy et al. präsentieren eine Sammlung technologischer Ansätze zur Begegnung der Anforderungen von Cyber-Physischen Systemen [35].

Die zentralen Herausforderungen in der Entwicklung von Cyber-Physischen Systemen gehen auf die durch Lee [89] beschriebenen komplementären Ansätze *cyberize the physical* und *physicalize the cyber* zurück. Der *cyberize the physical* Ansatz beschreibt die Herausforderung, physische Objekte im *Cyberspace* zu repräsentieren. Das Konzept *physicalize the*

cyber adressiert die Anpassung der Software an die physikalischen Prozesse, um insbesondere das Verhalten von Systemen besser virtuell beschreiben zu können. Dabei nimmt die Integration der Zeitabhängigkeit in die Software eine zentrale Rolle ein [89]. Zur Begegnung der steigenden Multidisziplinarität und Komplexität der Entwicklung von Cyber-Physischen Systemen, schlagen GRIMM ET AL. einen modellbasierten Ansatz vor [60]. Die Autoren beschreiben Qualitätsmerkmale und stellen eine schichtweise und abstrakte Struktur Cyber-Physischer Systeme vor. Eine Übersicht wissenschaftlicher Ansätze zur Entwicklung Cyber-Physischer Systeme bietet HEHENBERGER ET AL. [63]. Bestehende mechatronische Systeme haben eine lange Nutzungsdauer und müssen zu Cyber-Physischen Systemen weiterentwickelt werden. KUTSCHER ET AL. schlagen einen Ansatz zur Aufrüstung von bestehenden Systemen vor, indem eine Weiterentwicklungsmethode sowie ein Bewertungsschema zur Einschätzung des Ist- und Sollzustandes vorgestellt wird [84].

Der Einsatz Cyber-Physischer Systeme deckt ein breites Spektrum an Anwendungsfeldern in verschiedenen Branchen, wie der Mobilitäts- und Energiewirtschaft, der Medizin, dem Maschinenbau und der Automatisierungstechnik ab [91, 52, 2]. Für die Verbindung Cyber-Physischer Systeme mit der industriellen Produktion hat sich der Begriff *Cyber-Physical Production System* (CPPS) etabliert. Cyber-Physische Produktionssyteme unterscheiden sich hinsichtlich ihrer Eigenschaften nicht grundlegend von Cyber-Physischen Systemen. Durch ihren speziellen Anwendungskontext in der Produktion ergeben sich jedoch besondere Anforderungen [11]. So spielt beispielsweise die Fähigkeit, sich von unvorhergesehenen Störungen und Ausfällen von Systemteilen zu erholen, die so genannte Resilienz, eine zentrale Rolle [164]. Diese Störungen und Ausfälle sind z.B. durch die globale Vernetzung [161], den Einsatz von Funktechnologien [148], undefinierte Umweltbedingungen nicht vorhersehbar und erfordern neue Entwicklungsmethoden für Cyber-Physische Systeme [62].

2.2.2 Verwaltungsschale

Ein grundlegendes Konzept von Industrie 4.0 ist die sogenannten Industrie 4.0 Komponente (I4.0-Komponente). Gemäß DIN SPEC 91345 ist die I4.0-Komponente ein "weltweit eindeutig identifizierbarer, kommunikationsfähiger Teilnehmer bestehend aus Verwaltungsschale und Asset mit digitaler Verbindung innerhalb eines I4.0-Systems, welche dort Dienste mit definierten Quality of Service (QoS)-Eigenschaften anbietet“ [45]. Ein Asset ist dabei ein beliebiges Objekt aus der physischen oder der Informationswelt, das einen Wert für die Organisation besitzt [45]. Der Kontext der Verwaltungsschale wird in dem sogenannten *Referenzarchitekturmodell Industrie 4.0* dargestellt [45]. Mittels der Verwaltungsschale wird das Asset in die virtuelle Welt abgebildet und von jener repräsentiert, siehe Abbildung 2.3. Die Verwaltungsschale enthält Submodelle, welche die Informationen

und Funktionen des Assets beschreiben, zu welchen auch die Merkmale, Eigenschaften, Charakteristika, Zustände, Messdaten und Fähigkeiten gehören [112]. Die Verwaltungsschale ermöglicht damit die Interoperabilität der Assets unterschiedlicher Herkunft, indem standardisierte Kommunikationsschnittstellen und Anwendungen unterstützt werden. Die grundsätzliche Struktur der Verwaltungsschale ist per Metamodell [30] beschrieben, welches im veröffentlichten Dokument der Plattform Industrie 4.0 [28] vorgestellt wurde. Darauf aufbauend erschien in [29] die Vorlage für die Umsetzung der Schnittstellen der Verwaltungsschale. Bei der Implementierung der Verwaltungsschale werden standardisierte Technologien wie XML (*Extensible Markup Language*), JSON (*JavaScript Object Notation*) und OPC UA eingesetzt [112]. Beispielhafte Prototypen sind dabei der *AASX package explorer* [65] und die Umsetzungen unterschiedlicher Hersteller [68].

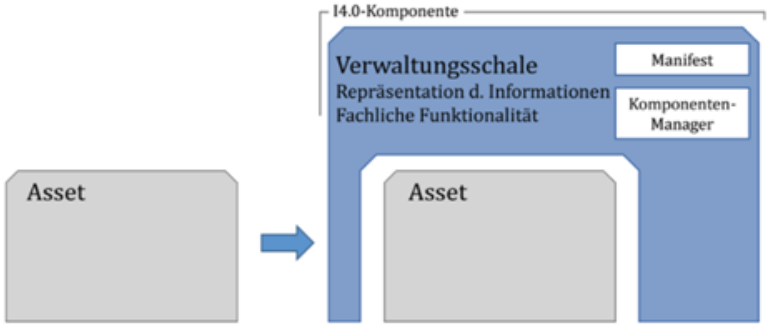

Abbildung 2.3: Erweiterung eines Assets zu einer I4.0-Komponente mittels einer Verwaltungsschale [45]

2.3 Digitaler Zwilling

Das Konzept des Digitalen Zwillings bildet die Grundlage der vorliegenden Dissertation und wird im Folgenden detailliert vorgestellt. Dazu erfolgt eine Analyse und strukturierte Darstellung des aktuellen Stands der Technik und Forschung. Zunächst wird einleitend die Grundidee des Digitalen Zwillings dargestellt und die im Rahmen dieser Dissertation gültige Definition ausgearbeitet. Anschließend werden einige Charakteristika eines Digitalen Zwillings aufgearbeitet. Darauf aufbauend erfolgt die Untersuchung wissenschaftlich vorgestellter Architekturen, die sich beim Design und Generierung eines Digitalen Zwillings einsetzen lassen. Der thematische Fokus bei der Generierung liegt dabei auf der Abbildung des Verhaltens des Digitalen Zwillings. Der tatsächliche Mehrwert eines Digitalen Zwillings

eröffnet sich insbesondere in der Nutzung, welche erst durch die Bereitstellung an einen Kunden bzw. den Nutzer ermöglicht wird.

2.3.1 Einleitung und Definition

Die Bezeichnung *Digitaler Zwilling* beschreibt die Vision einer digitalen Repräsentation eines physischen Produkts und eine Schlüsseltechnologie der Industrie 4.0. Die Konzepte rund um den Digitalen Zwilling haben in den letzten Jahren eine Vielfalt erreicht, die sich nach dem aktuellen Stand nicht in einer Definition vereinheitlicht lässt [9]. Aufgrund der Bedeutung in der vorliegenden Dissertation, werden daher zunächst die Historie und einige wichtige Eigenschaften des Digitalen Zwillings adressiert und eine für die Dissertation gültige Definition formuliert.

Die Grundidee der Verwendung von präzisen Abbildungen von Systemen unter Berücksichtigung des aktuellen Zustands entstand in der Raumfahrt. Während des Apollo-Raumfahrtprogramm der National Aeronautics and Space Administration (NASA) in den 60er und 70er Jahren wurden umfangreiche Simulatoren (siehe Abbildung 2.4) verwendet, um die Systemkomponenten von Raumfahrzeugen unter verschiedenen Rahmenbedingungen zu testen und die Astronauten für die Weltraummission vorzubereiten. Während der Mission wurden die Simulatoren außerdem eingesetzt, um den Zustand und die Umgebungsbedingungen zu simulieren und Anpassungen der Parameter zu testen.

Später entwickelte die NASA das Konzept unter der Bezeichnung Digitaler Zwilling (engl.: *Digital Twin*) weiter und definiert es wie folgt:

„Digital twin is an integrated multiphysics, multiscale simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin“ [136].

Grieves stellte im Jahr 2002 sein Konzept eines *Product-Lifecycle-Managements* industriellem Publikum vor, in welchem jedes System im Detail als zwei Systeme zu betrachten ist: das physische System selbst und ein zusätzliches virtuelles System, welches Informationen des physischen Systems beinhaltet und es damit virtuell spiegelt oder repräsentiert. Im Sinne des Product-Lifecycle-Managements sind die beiden Systeme durch Datenfluss in beide Richtungen miteinander verbunden. Nachträglich entspricht das Konzept bereits der Grundidee eines Digitalen Zwillings. Der Begriff *Digitaler Zwilling* wird jedoch erstmals als Synonym für das eigentliche Konzept eines *Information Mirroring Models* in einer gemeinsamen Veröffentlichung von Grieves und Vickers im Jahr 2011 publiziert [58]. Die Prägung des Begriffs wird von den Autoren in späteren Publikationen aufgearbeitet [59]. Grieves und Vickers definieren den Digitalen Zwilling wie folgt:

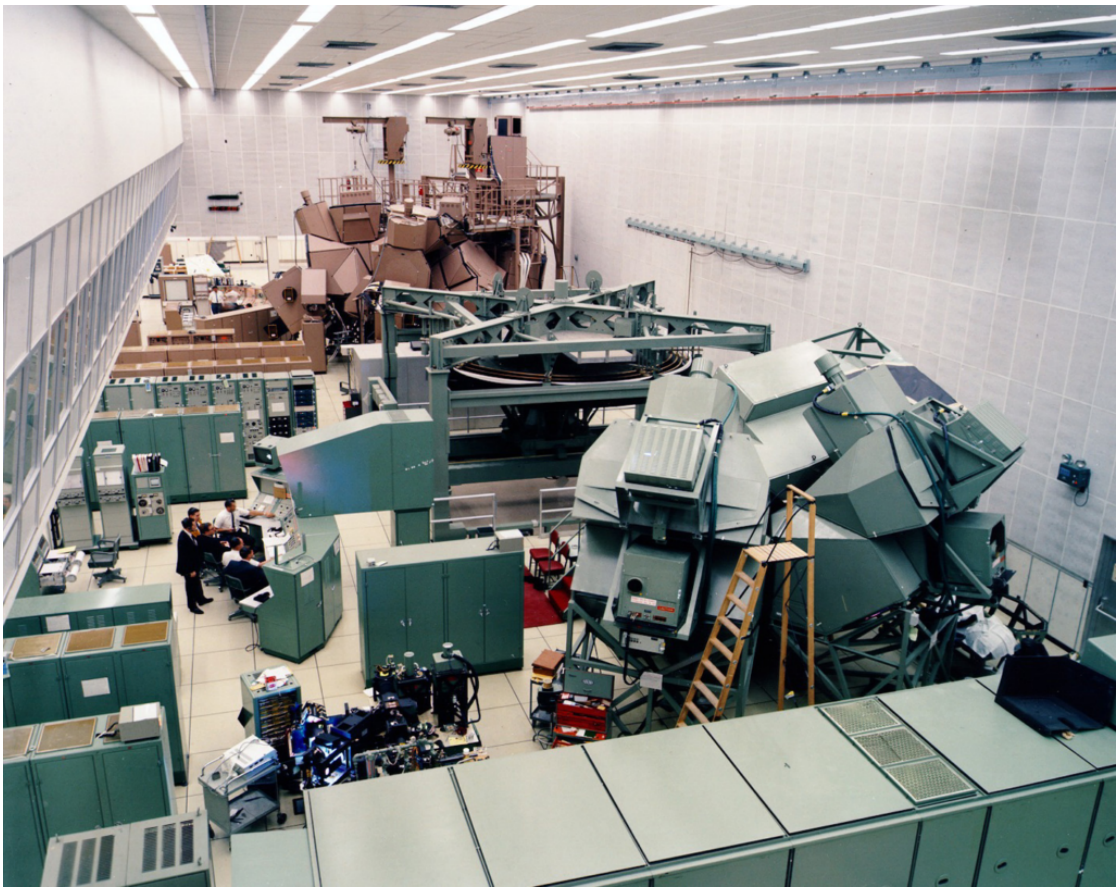

Abbildung 2.4: NASA Simulatoren als Abbildung der Raumfahrzeuge [Quelle: NASA]

„Digital Twin (DT) — the Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin. Digital Twins are of two types: Digital Twin Prototype (DTP) and Digital Twin Instance (DTI). DT's are operated on in a Digital Twin Environment (DTE)“ [59].

In der deutschen Forschungsgemeinschaft, die sich unter anderem in der *Wissenschaftlichen Gesellschaft für Produktentwicklung* (WiGeP) organisiert, wurde eine Definition des

Digitalen Zwillings konsolidiert:

„Ein Digitaler Zwilling ist eine digitale Repräsentation einer Produktinstanz (reales Gerät, Objekt, Maschine, Dienst oder immaterielles Gut) oder einer Instanz eines Produkt-Service-Systems (eines aus Produkt und zugehöriger Dienstleistung bestehenden Systems). Diese digitale Repräsentation beinhaltet ausgewählte Merkmale, Zustände und Verhalten der Produktinstanz oder des Systems. Ebenfalls werden innerhalb dieser digitalen Repräsentation während verschiedener Lebenszyklusphasen unterschiedliche Modelle, Informationen und Daten miteinander verknüpft“ [143].

Die WiGeP-Definition weitet das Konzept des Digitalen Zwillings auf immaterielle Güter und insbesondere auch auf gesamte Produkt-Service-Systeme aus. Der Definition ist zu entnehmen, dass auch das Verhalten eines Systems über verschiedene Lebenszyklusphasen durch den Digitalen Zwilling repräsentiert werden soll. Grieves und Vickers beschreiben ebenfalls einige Anwendungsfälle aus unterschiedlichen Produktlebensphasen [59]. Angefangen bei der Entwicklungsphase können nicht nur Form, sondern auch Verhalten virtuell modelliert werden, um Ressourcen, wie Zeit und Kosten für physische Prototypen, einzusparen. Weiter kann in der Produktion der Digitale Zwilling herangezogen werden, um die Herstellungsprozesse virtuell auszulegen, zu testen und zu optimieren. Dadurch werden Ressourcen eingespart und insbesondere bessere Lösungen gefunden, da das Testen von Varianten weniger Kosten verursacht im Vergleich zum physischen Prozess. In der Nutzungsphase ermöglicht der Digitale Zwilling ein besseres Verständnis des Systems und führt zum effektiveren und effizienteren Betrieb. Dies kann insbesondere genutzt werden, um unerwünschtes Verhalten zu reduzieren oder ganz auszuschließen [59]. Die während der Nutzung gesammelten Daten können außerdem verwendet werden, um diese zwischen den Digitalen Zwillingen auszutauschen und damit eine Gesamtoptimierung durchzuführen oder um nachfolgende Produktversionen zu verbessern. Schließlich können in der Entsorgungsphase Informationen aus dem Entwicklungsprozess genutzt werden, um eine sichere und umweltschonende Entsorgung zu ermöglichen [59] oder um die Wiederverwertung von Systemen und Komponenten zu unterstützen [42].

2.3.2 Charakteristika

Die Charakteristika eines Digitalen Zwillings hängen wesentlich von dem betrachteten physischen System und den damit verbundenen Anforderungen ab. Durch das breite Spektrum an Einsatzmöglichkeiten sind unterschiedlichste Arten eines Digitalen Zwillings möglich, die typischerweise jedoch in einigen Eigenschaften übereinstimmen. Kutscher et. al. zeigten im Zusammenhang mit dem *web-basierten Digitalen Zwilling* bereits die

sechs typischen Charakteristika auf: *Repräsentation von Daten*, *Repräsentation des Verhaltens*, *bidirektionale Verbindung*, *standardisierte externe Schnittstellen*, *standardisierte interne Schnittstellen* und *Servicequalität* [85]. Im Folgenden werden diese Charakteristika genauer erläutert und um zusätzliche Eigenschaften erweitert, die in der wissenschaftlichen Literatur [71, 46, 144] beschrieben sind und die insbesondere für das Konzept dieser Dissertation und dessen Validierung bedeutsam sind.

Repräsentation von Daten: Der Digitale Zwilling erfasst, speichert und stellt Daten zur Verfügung. Zu diesen Daten gehören aktuelle Zustandsdaten, die mittels Sensoren erfasst werden aber auch weitere Daten, die während des Produktlebens entstehen. Die Abbildung von statischen Informationen, wie beispielsweise die geometrische Beschreibung oder Materialeigenschaften gehört ebenfalls zu den Funktionen des Digitalen Zwillings. In diesem Zusammenhang ist Datenmanagement eine Funktion des Digitalen Zwillings [134], die aufgrund neuer Anforderungen an den Digitalen Zwilling angepasst werden muss [125]. Die Repräsentation individueller Daten einer Produktinstanz wird auch im Kontext des Produkt-Avatars [72] und unter der Bezeichnung Digitaler Schatten [135] erforscht.

Datenverarbeitung: Der Digitale Zwilling ist nicht nur für die Akkumulation und Bereitstellung der Daten zuständig, sondern diese Daten werden auf unterschiedliche Weise verarbeitet, um neue Informationen daraus zu generieren. Dies kann beispielsweise genutzt werden, um Vorhersagen über das System zu treffen [92]. Zur Datenverarbeitung können auch Simulationen verwendet werden, die einen wesentlichen Teil des Digitalen Zwillings ausmachen [31].

Repräsentation des Verhaltens: Eine wesentliche Charakteristik des Digitalen Zwillings ist die Repräsentation des Verhaltens des physischen Systems in Form von Zustandsänderungen über Zeit. Die Veränderungen des Systems können sowohl vergleichsweise schnell während des Betriebs erfolgen, als auch über den gesamten Produktlebenszyklus betrachtet werden [77]. Das Verhalten eines Systems weist dabei unterschiedliche Eigenschaften auf. Es kann sich auf physikalische, chemische oder biologische Prozesse beziehen, welche beispielsweise in mathematischen Gleichungen oder analytischen Modellen und Lösungsalgorithmen ausgedrückt werden [85]. Weiterhin wird das dynamische Verhalten von software-gesteuerten Systemen durch die in der Steuerungssoftware hinterlegte Logik und die Elektrik/Elektronik bestimmt. Das System reagiert damit aktiv auf Eingaben oder veränderte Randbedingungen.

Bidirektionale Verbindung: Ein wesentliches Unterscheidungsmerkmal zwischen einem

Digitalen Zwilling und einer traditionellen Simulation ist die bidirektionale Verbindung zwischen der realen und virtuellen Welt [71]. Diese Verbindung liefert im Gegensatz zu einer Simulation sich ständig verändernde Eingaben, während eine klassische Simulation auf Basis eines festen Inputdecks initiiert wird und sich während der Simulation an den Eingangsdaten nichts mehr ändert. Die bidirektionale Verbindung muss dabei nicht zu jedem Zeitpunkt vollumfänglich ausgebildet sein, sondern es existieren Anwendungsfälle, in denen eine sofortige Synchronisation der Zwillinge nicht erwünscht ist, indem z.B. Tests ausschließlich am virtuellen Modell durchgeführt werden [83]. Dazu müssen entweder Teile der Verbindung unterbrochen sein oder es wird eine unabhängige und temporäre Instanz des Digitalen Zwillings erzeugt, um zukünftige Systemzustände auf Basis des aktuellen Zustands und mit variierenden Zustandsgrößen vorausschauend zu analysieren. Jones et al. heben jedoch hervor, dass ohne bidirektionale Verbindung der Mehrwert des Digitalen Zwilling gegenüber einem konventionellen Produktmodell nicht greifbar ist, da der Zustand des physischen Systems nicht beeinflusst werden kann [71]. Kutscher et al. haben eine modellbasierte und weitgehend standardisierte Schnittstelle entwickelt, um die bidirektionale Verbindung zwischen den Zwillingen zu ermöglichen [86].

Synchronisation der Zwillinge: Mittels der bidirektionalen Verbindung werden Daten übertragen, die unter anderem zur Synchronisation der Zwillinge genutzt werden. Diese Übertragung muss automatisch stattfinden [46]. Dieser Vorgang wird im Englischen als *Twinning* bezeichnet. Die Sensordaten des Physischen Zwillings werden dem Digitalen Zwilling zur Verfügung gestellt, um den aktuellen Zustand nachzubilden und umgekehrt fließen Informationen, die beispielsweise zur Steuerung des physischen Systems herangezogen werden. Die Geschwindigkeit der Übertragung spielt weiterhin eine bedeutsame Rolle. Jones et al. haben identifiziert, dass für die Synchronisationsgeschwindigkeit zwischen den Zwillingen in den analysierten wissenschaftlichen Arbeiten als *Echtzeit* angegeben wird [71]. Das bedeutet, dass eine Änderung im physischen System eine nahezu sofortige Änderung im Digitalen Zwilling hervorrufen würde [71].

Standardisierte interne Schnittstellen: Der Digitale Zwilling besteht aus Daten, Modellen und Software aus unterschiedlichen Domänen, die in einer modularen Struktur über standardisierte und herstellerunabhängige Schnittstellen miteinander verbunden sind [86]. Dadurch nimmt die Integration der einzelnen Module eine bedeutsame Rolle ein, da diese aus unterschiedlichen Domänen stammen, die konventionell keine Schnittstellen zueinander aufweisen und im Digitalen Zwilling erstmals Daten austauschen müssen. Gleichzeitig müssen die internen Schnittstellen eine Erweiterbarkeit und Flexibilität des Digitalen Zwillings sicherstellen [85].

Detailtreue: Ein weiteres wesentliches Merkmal des Digitalen Zwillings ist die Detailtreue (engl.: Fidelity). Mittels einer hohen Detailtreue kann beim Menschen erst der Eindruck entstehen, dass es sich bei den Zwillingen um ein und dasselbe System handelt. Nach Grieves und Vickers ist es sogar das Ziel eines Digitalen Zwillings, keinen vom Menschen erkennbaren Unterschied aufzuweisen [59]. Gemäß einer Untersuchung von Durao et al. stellt die Detailtreue eines der meist adressierten Charakteristika in der wissenschaftlichen Literatur dar [46] dar. Bei der Detailtreue muss mit den heutigen und in naher Zukunft verfügbaren Rechenressourcen stets ein Kompromiss gegenüber Geschwindigkeit getroffen werden. Mit steigender Granularität der Daten, Modelle und Simulationen benötigen die Berechnungen mehr Ressourcen sowie Zeit, sodass die Echtzeit nicht mehr sichergestellt werden kann [85]. Um einen Digitalen Zwilling in einem Anwendungsfall effizient einsetzen zu können, muss die Detailtreue flexibel einstellbar sein [124].

Skalierbarkeit: Um eine flexibel einstellbare Detailtreue zu ermöglichen, muss der Digitale Zwilling ausgeprägte Funktionen zur Skalierung der Ressourcen aufweisen. Die Skalierbarkeit kann sich auf die Datenspeicherung, Rechenressourcen und andere Funktionen des Digitalen Zwillings beziehen [4]. Plesker et al. entwickelten als Teil des web-basierten Digitalen Zwillings [85] eine Methode zur flexiblen Konfiguration und skalierbaren Verteilung der Ressourcen auf externe Dienstleister in Form von Simulationsservern und Cloud-Plattformen [124].

Konnektivität und Kommunikation: Der Digitale Zwilling steht nicht nur mit dem physischen System in bidirektionaler Verbindung, sondern es werden auch Daten aus der Umwelt und von externen Kommunikationspartnern bezogen [126]. Hierfür wird Konnektivität in Form von externen Schnittstellen benötigt, die eine standardisierte Kommunikation ermöglicht [85]. Für die globale Kommunikation ist es grundlegend, dass eine eindeutige Identifikation möglich ist [134]. Unter Konnektivität mittels externer Schnittstellen wird auch die Mensch-Maschine-Schnittstelle berücksichtigt, welche in der Interaktion mit dem Menschen eine zentrale Rolle einnimmt [134].

Autonomie: Der Digitale Zwilling wird als ein wesentliches Element der Autonomisierung von Systemen betrachtet [126]. Durch die Akkumulation von Informationen zum Produkt, zum Prozess und zur Ressource, die für selbstständige Entscheidungsfindung technischer Systeme notwendig ist, kann die Autonomie auf ein neues Niveau gehoben werden [126].

Servicequalität: Der Nutzen eines Digitalen Zwillings kann erst zum Tragen kommen, wenn dieser zuverlässig und mit definierter Servicequalität bereitgestellt werden kann. Insbesondere im industriellen Umfeld müssen definierte Grundvoraussetzungen beispiels-

weise hinsichtlich der Qualität der Daten [150], Übertragungs- und Synchronisationsgeschwindigkeit [163], Ausfallsicherheit usw., sichergestellt werden.

Sicherheit: Aufgrund der globalen Vernetzung muss die Sicherheit der Daten, Sicherheit der Kommunikationsschnittstellen und die damit verbundene Sicherheit des Betriebs der Systeme gewährleistet sind. Diese Sicherheitsthemen werden bereits in der wissenschaftlichen Arbeit rund um Industrie 4.0 erforscht [18] und gewinnen mit dem Digitalen Zwilling zusätzliche Bedeutung. Der Digitale Zwilling selbst benötigt zwar einerseits eine umfassende Absicherung, kann andererseits auch verwendet werden, um innovative Sicherheitskonzepte zu verwirklichen [76].

Die beschriebenen Charakteristika wurden mit dem Fokus der Dissertation und unter der Annahme, dass es sich um ein physikalisches System aus der Produktion handelt, ausgewählt. In der wissenschaftlichen Literatur werden weitere Charakteristika benannt [46].

2.3.3 Architektur und Schnittstellen

Die Festlegung der Architektur als einen abstrakten Plan der Umsetzung, ist eine der ersten Entscheidungen bei der Entwicklung des Digitalen Zwillings und ist sehr weittragend. Durch die diversen Lebensphasen des zu entwickelnden Systems, müssen die bedeutsamen Faktoren aus jeder Lebensphase berücksichtigt werden. Diese Faktoren sind in der Entwicklungsphase zum Teil noch nicht bekannt und entstehen erst in der Nutzungsphase. Demnach muss die Architektur eines Digitalen Zwillings die notwendige Flexibilität aufweisen, um diesen effektiv nutzen und weiterentwickeln zu können. Im Folgenden werden einige Ansätze gemäß wissenschaftlicher Literatur zur Auslegung der Architektur beschrieben.

Schleich et al. stellen ein *Referenzmodell des Digitalen Zwillings* als konzeptionelle Basis und Vorlage für die Implementierung eines Digitalen Zwillings vor [128]. Das Referenzmodell basiert auf der Modellierung von Geometrien und berücksichtigt Charakteristika wie Skalierbarkeit, Interoperabilität, Erweiterbarkeit und Genauigkeit [128].

Schluse et al. entwickeln einen *experimentierbaren Digitalen Zwilling* auf Basis der Industrie 4.0- und Simulationstechnologien, welcher unterschiedliche Modelle und Simulationen miteinander verbindet, um einen erfahrbaren Digitalen Zwilling umzusetzen [129]. Die Basisstruktur des experimentierbaren Digitale Zwillings besteht aus vier simulierten Elementen: Mensch-Maschine-Schnittstelle, Datenverarbeitungssystem, Kommunikationsinfrastruktur und technisches Asset. Auf Basis dieser Architektur kann der experimentierbare Digitale Zwilling für simulationsbasiertes Engineering, Optimierung und Steuerung eingesetzt werden [129].

Damjanovic-Behrendt und Behrendt verwenden Open Source Software, um einen *Demonstrator eines Digitalen Zwillings im Kontext von Smart Manufacturing* zu entwickeln [41]. Die zwei Hauptkomponenten dieses Konzepts des Digitalen Zwillings sind ein *Virtualization Manager* und ein *Interoparability Manager*. Unter dem *Virtualization Manager* werden Funktionen zum Management von *Daten*, *Modellen* und *Services* zusammengefasst. Der *Interoperability Manager* enthält wiederum einen *Monitoring-*, *Decision-Making* und *Simulation-Manager*. Die Architektur und die Schnittstellen zielen auf die Simulation, Analyse und Vorhersage von Ereignissen im Produktionsökosystem.

Rovere et al. stellen eine Referenzarchitektur mit der Bezeichnung *Centralized Support Infrastructure* auf Basis von Microservices vor [40]. Die Autoren verwenden ein im Rahmen des europäischen *MAYA* Projekts entwickeltes System, um die Synchronisation der Cyber-Physischen Systeme der gesamten Produktion sowie der Digitalen Zwillinge zu ermöglichen. Dazu werden mittels einer Web-Technologie die Daten der Produktionssysteme analysiert und Simulationsmodelle und -ergebnisse zwischen Simulatoren ausgetauscht [40].

Zheng und Sivabalan stellen einen generischen Ansatz zur Umsetzung eines *Digitalen Zwillings eines Cyber-Physischen Systems* vor [165]. Darin wird auf Basis einer vierschichtigen Architektur des Cyber-Physisches System eines Digitaler Zwilling bestehend aus den drei Bereichen *Digitales Modell*, *Berechnungsmodell* und *Graphen-Basiertes Modell* generiert. Der so generierte Digitale Zwilling simuliert zeitgleich das physische Verhalten der realen Welt und die Charakteristika des Digitalen Zwillings [165].

Schroeder et al. stellen eine *Referenzarchitektur des Digitalen Zwillings* vor und beschreiben den Zusammenhang der einzelnen Komponenten [134]. Der Beschreibung zufolge beinhaltet ein Digitaler Zwilling mindestens ein *physikalisches Modell*, welches die physikalische Komponente beschreibt. Weiterhin muss es mindestens ein *Application Programming Interface* (API) aufweisen, welches einen definierten Zugriff auf die Funktionen des Digitalen Zwillings erlaubt. Dagegen sind die Komponenten *Speicher*, *Events*, *Methoden*, *Zugriffskontrolle* und *Mensch-Maschine-Schnittstelle* optional [134].

Weitere wissenschaftliche Literatur behandelt spezielle Systeme und Technologien, wie beispielsweise ein Architektur-Referenzmodell für Digitale Zwillinge cloudbasierter Cyber-Physischer Systeme [4] oder eine Digitaler Zwilling Architektur, die auf Internet Of Things Technologien basiert [141].

Ein Teil einer wirksamen Architektur eines zu entwickelnden Systems sind die Verbindungspunkte zwischen den einzelnen Elementen innerhalb des Systems und die Schnittstellen zu externen Systemen. Die Schnittstellen zwischen einzelnen Funktionen, welche im realen System auch einen Übergang zwischen unterschiedlichen Disziplinen darstellen können, nehmen im behandelten Kontext eine besondere Rolle ein. In einer Produktent-

wicklungsphase generierte Schnittstelle wird in der nächsten Phase durch eine andere Domäne genutzt. Mangelndes Verständnis dieser Schnittstelle kann jedoch in unvorhergesehenem und unerwünschtem Verhalten des Gesamtsystems resultieren [59]. Aus diesem Grund müssen zwischen-disziplinäre Schnittstellen im Digitalen Zwilling mit besonderer Sorgfalt erstellt werden. Neben den internen Schnittstellen, ist es eine wesentliche Charakteristik des Digitalen Zwillings, eine bidirektionale Verbindung zum physischen Objekt herzustellen. Der Datenaustausch zwischen dem digitalen und physischen Objekt besteht dabei während des gesamten Produktlebenszyklus [81]. Ein notwendiges Merkmal der Schnittstelle zwischen dem digitalen und physischen Objekt ist die Semantik der Daten. Durch eine informationsmodell-basierte Schnittstelle kann zwischen den Objekten eine semantische Verbindung geschaffen werden, die bei gleichen oder ähnlichen Systemen wiederverwendbar ist, ohne ressourcenintensiver Individuallösungen zu bedürfen. Diese Grundidee einer semantischen Schnittstelle wurde bereits im Konzept der Verwaltungsschale aufgegriffen, indem die darin umgesetzte Verbindungen zwischen einzelnen Systemen auf semantischen Netzwerken basiert [123]. Eine mögliche Technologie zur Umsetzung einer semantischen Schnittstelle im Digitalen Zwilling bietet OPC UA. Dadurch können Cyber-Physische Produktionssysteme mit den jeweiligen Digitalen Zwillingen synchronisiert werden und aktiv Daten bereitstellen, die durch die integrierte Semantik besonders auch für Simulationen eingesetzt werden können [15]. Kutscher et al. beschreiben ein Konzept zum Einsatz von OPC UA als modellbasierte, zentrale Schnittstelle zwischen einem Digitalen und Physischen Zwilling [86].

2.3.4 Design

Die Entwicklung Digitaler Zwillinge auf Basis einer Referenzarchitektur stellt eine zentrale Forschungsfrage dar. Dabei existieren unterschiedliche Ansätze, die auf verschiedenen Ausgangszuständen basieren und auch unterschiedliche Ziele anvisieren. Grundsätzlich können Digitale Zwillinge bereits während der Produktentwicklung berücksichtigt werden und gemeinsam mit den physischen Zwillingen instanziiert oder in einer späteren Produktlebensphase nachträglich generiert werden. Eine Möglichkeit der Entstehung des Digitalen Zwillings aus dem *Digitalen Master* der Produktentwicklungsphase beschreibt die WiGeP in einem Positionspapier [143]. Demnach enthält der Digitale Zwilling Verknüpfungen aus dem Digitalen Master und Digitalem Schatten. Der Digitale Master stellt dabei die Produktgeometrie und verhaltensbeschreibende Modelle bereit, während der Digitale Schatten die Betriebs- Zustands- oder Prozessdaten der Produktinstanz abbildet [143]. Vorarbeiten zu diesem Konzept finden sich auch in der Forschung von Stark et al. [145] wieder. Falls das Produkt bereits instanziiert ist, kann der Digitale Zwilling nachträglich generiert werden. Dazu erarbeiteten Kutscher et al. im Rahmen des Teilprojekts

Software-Factory 4.0: Reengineering für die Industrie 4.0 ein Vorgehen, um Bestandssysteme zu Cyber-Physischen Systemen weiterzuentwickeln und diese mit einem Digitalen Zwilling auszustatten [84]. Einige Autoren machen keinen grundsätzlichen Unterschied zwischen der simultanen Entstehung beider Zwillinge und nachträglichen Bereitstellung der Digitalen Zwillings, sondern beschreiben generische Methoden, die in beiden Fällen angewendet werden können. Luo et al. beschreiben eine Methode zur Modellierung Digitaler Zwillinge von CNC Systemen, die aus einem intelligenten Modell zur Analyse von Daten und einem beschreibenden Modell, umgesetzt mittels OPC UA zur Kommunikation zwischen den Zwillingen, bestehen [94]. Liu et al. setzten die Technologie *MTConnect* als eine Alternative zu OPC UA ein, um Systeme mit den jeweiligen Digitalen Zwillingen zu verbinden [93]. Schroeder et al. generieren mittels *AutomationML* eine virtuelle Repräsentation eines technischen Gerätes und modellieren mittels dieser Sprache auch den Digitalen Zwilling [134]. Aufgrund des Umfangs der Daten und Modelle in einem Digitalen Zwilling, ist es zielführend, die Generierung möglichst zu automatisieren. Diesem Ziel widmet sich die Veröffentlichung von Haag und Anderl, in der ein Konzept zur automatisierten Generierung geometrischer Repräsentanz für einen Digitalen Zwilling beschrieben wird, die sich durch den Einsatz von *STEP* (*STandard for the Exchange of Product model data)* von der Maßen des tatsächlich gefertigten Bauteil ableitet [61].

2.3.5 Verhaltensabbildung

Im Digitalen Zwilling können Daten aus allen Produktlebensphasen genutzt werden, um Informationen zu gegenwärtigen, vergangenen und zukünftigen Zuständen und Ereignissen zu gewinnen. Um aus den vorliegenden Daten neue Informationen abzuleiten, werden unter anderem Modelle mithilfe von Werkzeugen in Form von Simulationen ausgeführt. Eine Simulation kann jeweils einen Teil des Digitalen Zwillings abbilden. Jones et al. klassifizierten zehn Eigenschaften des Digitalen Zwillings. Diese sind *Form, Funktionalität, Verschleiß, Standort, Prozess, Zeit, Zustand, Leistung, Umgebung* und *sonstige qualitativen Parameter* [71]. Für jeden der Eigenschaften können Simulationen eingesetzt werden, um aus Daten nicht-triviale Erkenntnisse zu gewinnen. Boschert und Rosen sehen in dem Digitalen Zwilling die nächste Evolutionsstufe der Simulationstechnologie und beschreiben in diesem Zusammenhang die für diese Sichtweise wesentlichen Eigenschaften wie *Nutzen, Architektur, Lebenszyklus* und *Wertschöpfungsketten* [31]. Dabei sind die Simulationen im Digitalen Zwilling nicht isoliert und diskret zu betrachten, indem ein Simulationsdurchlauf auf Basis eines einzelnen Inputdecks durchgeführt und anschließend vom Menschen ausgewertet wird, sondern als eine Verkettung mehrerer Simulationen, die autonom und kontinuierlich stattfinden. Durch die unüberwachte Simulationstätigkeit muss der Digitale Zwilling die Ergebnisse der Simulation interpretieren können, um bei Bedarf selbstständig

Funktionen, wie beispielsweise ein Nothalt, auslösen zu können. Die Verknüpfung von Simulationen zu Simulationsketten bildet eine sogenannte Co-Simulation [54], die es erfordert, Simulationen aus teilweise unterschiedlichen Domänen mittels gemeinsamer Schnittstellen miteinander zu verbinden.

Der Fokus der vorliegenden Dissertation liegt auf der Abbildung der Verhaltenslogik eines Digitalen Zwillings. Als Grundlage werden im Folgenden unterschiedliche Ansätze beschrieben, das Verhalten eines Systems im Digitalen Zwilling zu berücksichtigen. Das Verhalten eines Systems kann aus unterschiedlichen Perspektiven betrachtet werden. So können beispielsweise die Spannungsverläufe oder Deformationen eines Systems aufgrund einer externen Kraft analysiert werden. Es können thermische, aerodynamische, akustische oder weitere Simulationsarten durchgeführt werden, um zu untersuchen, wie das System auf Einflussgrößen wie Temperatur, Geschwindigkeit, Schall usw., reagiert. In einem konventionellen Simulationsverfahren werden Parameter, die für die jeweilige Simulationsart interessant sind, aus dem Produkt abstrahiert und in die Simulationskonfiguration einbezogen. Nach Durchführung der Simulation werden die Ergebnisse interpretiert und die Erkenntnisse ggf. wieder in das Produkt zurückgeführt. Das Ziel des Digitaler Zwilling Konzepts ist es jedoch, die Simulationen zu integrieren, um einerseits den Umweg über *Abstraktion - Simulation - Interpretation* zu vermeiden und insbesondere die Simulationen über die Entwicklung hinaus in allen Phasen des Produktlebenszyklus zu nutzen, um beispielsweise unerwünschtes und unvorhergesehenes Verhalten zu vermeiden oder zumindest zu reduzieren [59]. In der Literatur wird betont, dass der Digitale Zwilling auch das Verhalten eines Systems abbilden muss. Stark et al. beschreiben in Ihrer Arbeit ein Verhaltensmodell, welches aus den vier Elementen *kinematisches Verhalten*, *logisches Verhalten*, *Verhandlung und Ausführung von Fertigkeiten* und *physikalisches Verhalten* zusammengesetzt ist [145]. Die Teilmodelle zur Abbildung des Verhaltens entstehen während der Entwicklung aus dem Design der Mechanik, der Elektrik und aus der Software. Das so entstehende multi-disziplinäre Modell kann anschließend für verschiedene Anwendungsfälle, z.B. als immersives Erlebnis in virtueller Realität oder zum Testen von PLC (*Programmable Logic Controller*) Programmen, verwendet werden. Das Verhaltensmodell kann außerdem für virtuelles Prototyping und Validierung von Cyber-Physischen-Produktions-Systemen herangezogen werden [145]. Grieves und Vickers schlagen vor, das Konzept des Digitalen Zwillings einzusetzen, um unvorhergesehenes und unerwünschtes Verhalten vorherzusagen, zu reduzieren oder ganz zu beseitigen [59]. Gemäß ihrer Forschungsarbeit unterteilt sich das Verhalten eines Systems in vorhergesehenes und unvorhergesehenes Verhalten, wobei diese jeweils erwünscht oder unerwünscht sein können, siehe Abbildung 2.5 [59].

Um Vorhersagen über das Verhalten treffen zu können, ist es wesentlich, das System möglichst genau zu verstehen und die komplexen Zusammenhänge nachzuvollziehen.

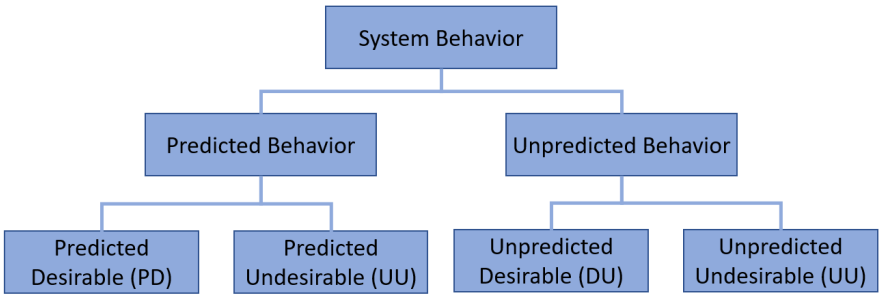

Abbildung 2.5: Verhaltenskategorien eines Systems nach Grieves und Vickers [59]

Dies kann gelingen, indem Verhaltenseigenschaften des Systems im Digitalen Zwilling abgebildet werden. Grieves und Vickers weisen auch auf die drei Haupthürden hin, die zur Realisierung des Konzepts eines Digitalen Zwillings überwunden werden müssen. Diese Hürden resultieren aus *Datensilos*, *mangelndem Verständnis der physikalischen Welt* und der *Zahl an möglichen Zuständen*, die ein System einnehmen kann [59].

Einen weiteren Ansatz zur Untersuchung der Interaktion zwischen Mechanik und Software eines mechatronischen Systems entwickelte Schöfer mittels eines Konzepts zur Absicherung von Steuergerätesoftware in mechatronischen Produkten durch Verknüpfung des Software-Modells mit Geometrie-Modellen in einer gekoppelten Model-in-the-Loop Simulation. Mit dem entwickelten Vorgehen kann insbesondere in frühen Phasen der Produktentwicklung die Korrektheit der Reaktion der Software auf geänderte Randbedingungen überprüft werden [131]. Das Konzept bildet keine zeitlichen Aspekte und Echtzeitbedingungen ab und betrachtet nicht das vollständige dynamische mechatronische Systemverhalten. In diesem Konzept wird außerdem ein Modell der Steuerungssoftware benötigt und es kann nicht der Code der eingebetteten Software verwendet werden.

Um zu bewerten, wie gut der Digitale Zwilling das Verhalten des physischen Systems nachbilden kann, muss die Verhaltensabbildung evaluiert und bewertet werden. Hierzu hat Grieves das Testverfahren *Grieves' Test of Virtuality* (GTV) entwickelt, um zu bewerten, wie genau ein virtuelles System das physische Gegenstück nachbildet [59]. Bei der Durchführung des Testverfahrens beobachtet ein Mensch zwei Bildschirme. Ein Bildschirm zeigt den physischen Raum, während der zweite den virtuellen Raum abbildet. Anschließend können drei Tests durchgeführt werden: ein *Sensorisch Visueller Test*, ein *Performanztest* und ein *Reflektivitätstest*. Der Sensorisch Visuelle Test zielt dabei auf die statische Abbildung des physischen Objekts. Beim Performanztest wird die Äquivalenz im Verhalten beurteilt, indem z.B. das System betrieben wird. Der Reflektivitätstest dient der Bewertung des virtuellen Systems in Wechselwirkung mit der Umwelt, welche hierfür

ebenfalls abgebildet werden muss. Ein Test gilt als bestanden, wenn der menschliche Beobachter in den jeweiligen Disziplinen keinen Unterschied zwischen dem Realen und dem Virtuellen feststellen kann [59].

Die Gruppe der verhaltensabbildenden Modelle umfasst Eigenschaften eines Systems, welche die Reaktion auf eine beliebige Einflussgröße, wie z.B. eine Kraft oder Temperatur, beobachtet und modelliert werden können. Der Schwerpunkt der vorliegenden Dissertation liegt jedoch auf der Verhaltenslogik technischer Systeme, die in Form von Steuerungssoftware im System implementiert ist. Eine Definition der adressierten Verhaltenslogik folgt in Kapitel 4.2.

2.3.6 Bereitstellung Digitaler Zwillinge

Der Mehrwert eines Digitalen Zwillings entsteht in unterschiedlichen Szenarien in allen Produktlebensphasen und insbesondere während der Nutzungsphase. Eine Voraussetzung, die in der Forschung bisher wenig adressiert wird, ist die Bereitstellung des Digitalen Zwillings an den Kunden bzw. den eigentlichen Nutzer. Die Bereitstellung stellt für einige entwickelte Ansätze insofern eine besondere Herausforderung dar, weil für die Entwicklung oft proprietäre Werkzeuge verwendet werden, die im späteren Nutzungsumfeld nicht zur Verfügung stehen und meist nur unter enormen Mehraufwand einzuführen sind. Um das Konzept eines Digitalen Zwillings tatsächlich praxistauglich zu machen, muss die Bereitstellung bereits während der Entwicklung berücksichtigt werden. Ein möglicher Ansatz ist die Verwendung von standardisierten Modellierungssprachen. Durch den Einsatz standardisierter Modellierungssprachen können die Modelle des Digitalen Zwillings in unterschiedlichen, die konkrete Modellierungssprache unterstützende Umgebungen ausgeführt werden. Schroeder et al. stellen den in *AutomationML* (*Automation Modeling Language*) modellierten Digitalen Zwilling mittels eines Python Skripts und eines Web Servers bereit. Dabei kommen für die Kommunikation die nach REST formulierten HTTP (*Hypertext Transfer Protocol*) Anfragen zum Einsatz [133]. Eine weitere Möglichkeit der Bereitstellung Digitaler Zwillinge ist die Instrumentalisierung web-basierter Technologien. Die zahlreichen Technologien des World Wide Webs sind in der Regel offen, standardisiert und systemunabhängig. Dies schafft eine Grundlage für die praktische Anwendung und Verbreitung des Konzepts des Digitalen Zwillings [85]. Konstatinov et al. nennen drei Gründe für den Einsatz von web-basierter Technologien für die Implementierung des Digitalen Zwillings [78]. Erstens die *Konnektivität*, welche für schnelle Übertragung von Daten geeignet ist, zweitens die *Verfügbarkeit* der Infrastruktur web-basierter Anwendungen in Zusammenhang mit der Cloud-Technologie und drittens die *Geschäftsmodelle*, welche auf Marketing, Verteilung der Anwendungen, Rechte, Anwendersupport und zukünftige Entwicklung beruhen. Durch erhöhte Sichtbarkeit, Übertragbarkeit und Transparenz der

Daten wird eine schnelle Entscheidungsfindung unterstützt. Konstantinov et al. entwickelten einen web-basierten Digitalen Zwilling eines mittels PLC automatisierten Systems. Für die Visualisierung verwendeten Sie WebGL (Web Graphics Library) Technologien, welche die graphischen Visualisierungen mittels OPC UA mit einer PLC verband. Die Anwendung der entwickelten Architektur begrenzt sich auf Virtuelle Inbetriebnahme einer PLC mittels des Digitalen Zwillings, die Repräsentation des Zustandes des physischen Zwillings mittels des Webbrowsers und der Manipulation des PLC Programms bzw. die Programmierung der PLC [78]. Kutscher et al. setzten in ihrem Ansatz eines web-basierten Digitalen Zwillings durchgehend auf die Webtechnologie, indem die *Datenhaltung*, *verhaltensabbildenden Simulationen*, *Kommunikation*, *Schnittstellen* und *Visualisierungen* im Digitalen Zwilling umgesetzt werden und von einer internetfähigen Ressource mit einem Webbrowser erreicht werden können [85]. Damit kann der Digitale Zwilling flexibel bereitgestellt werden, indem der Nutzer vorher keine Infrastruktur aufbauen muss, sondern den Digitalen Zwilling mit in der Regel vorhandener Software- und Hardwareinfrastruktur gleich nutzen kann. Einige der bereits vorgestellten Arbeiten verfolgen eine ähnliche Strategie und setzten web-basierte Technologie für Teile des Digitalen Zwillings ein [132, 141, 93].

2.4 Simulation im Digitalen Zwilling

Das vorige Kapitel stellte das abstrakte Konzept des Digitalen Zwillings dar. Nun soll im Folgenden auf die Simulation der Verhaltenslogik im Digitalen Zwilling eingegangen werden. In der modernen Produktentwicklung wird die Simulationstechnologie bereits eingesetzt, um Entwicklungsaufgaben zu unterstützen und das System zu verifizieren und validieren. Auch in der Nutzungsphase werden Simulationen verwendet, um den Betrieb zu optimieren und Fehler vorherzusagen. Diese Maßnahmen unterstützen die Beherrschung der Komplexität mechatronischer Systeme [31]. Durch die Vernetzung der Systeme steigt die Komplexität jedoch weiterhin. Gleichzeitig besteht der Wunsch, das Verhalten eines Systems vorhersagen zu können. Im Nachfolgenden wird zunächst die Co-Simulation als die besondere Herausforderung im Digitalen Zwilling thematisiert. Anschließend werden einige Architektur- und Schnittstellenstandards adressiert, die im Rahmen der Co-Simulation zum Tragen kommen. Abschließend wird die Disziplin der Entwicklung und Simulation eingebetteter Software und Hardware mittels Virtueller Plattformen behandelt, da im Rahmen der Dissertation gerade diese Technologien in den Digitalen Zwilling aufgenommen werden.

2.4.1 Co-Simulation

Die Besonderheit in der Abbildung des Verhaltens mechatronischer und Cyber-Physischer Systeme in einem Digitalen Zwilling liegt darin, dass hierbei unterschiedliche Domänen miteinander integriert werden müssen. Die grundlegenden Domänen der Mechanik, der Elektrik/Elektronik, der Informatik und weitere Disziplinen, welche je nach Systemart z.B. die Aerodynamik, Thermodynamik oder hydraulische Komponenten betreffen, entwickeln und verwenden eigene Modelle, um die einzelnen Eigenschaften des Systems abzubilden. Um das ganzheitliche Verhalten virtuell zu repräsentieren, müssen die einzelnen Modelle in einer interdisziplinären Simulation gekoppelt werden [82]. Zur Erstellung und Durchführung von Multi-Domänen Simulationen existieren unterschiedliche Ansätze und Werkzeuge. Die erste Möglichkeit besteht darin, das gesamte System in einem Solver zu integrieren (engl.: *Model Coupling*). Bei diesem Verfahren müssen die Modelle aus den einzelnen Domänen die gleiche mathematische Formulierung aufweisen. Die zweite Option besteht in der Kopplung einzelner Subsysteme auf Basis gemeinsamer Schnittstellen (engl.: *Solver / Process Coupling*) [49, 82].

Der Kontext eines Digitalen Zwillings stellt zusätzliche Anforderungen an die domänenübergreifende Simulation. Die einzelnen Modelle sind nicht aus der gleichen Domäne und unterscheiden sich in der Modellierungssprache [54]. In einer dynamischen Simulationsumgebung können einzelne Teilnehmer der Simulation zur Laufzeit beitreten und wieder verlassen [73, 74], wobei diese weiterführende Anforderung zwar nicht Teil der Definition des Digitalen Zwillings, aber dennoch in einigen Anwendungsfällen bedeutsam ist. Aus diesem Hintergrund ist Modell-Kopplung wenig geeignet, sondern es muss eine Solver- und Prozess-Kopplung zur Laufzeit erfolgen. Damit können die bestehenden Simulationen und Modelle aus den einzelnen Domänen weiterverwendet werden, indem die spezialisierten Solver und Simulationswerkzeuge die einzelnen Berechnungen durchführen und als Simulationseinheiten einer Co-Simulation verbunden werden [49]. In der Co-Simulation muss jedoch einigen Herausforderungen, wie beispielsweise der zusätzlichen *zeitlichen Diskretisierung von Datenaustausch* zwischen den Simulationseinheiten, der *Synchronisierung der Simulationseinheiten* und allgemein der zusätzlich erforderlichen *Kommunikation* [49] begegnet werden.

Solver / Prozess Kopplung

Die Co-Simulation wird auch als *Solver / Prozess Kopplung* oder auch als *schwache Kopplung* bezeichnet. Bei dieser Kopplungsart müssen die einzelnen Modelle nicht zwangsweise die gleiche mathematische Formulierungen haben, sondern werden jeweils von eigenen, in der jeweiligen Domäne etablierten Solvern ausgeführt [49]. Die einzelnen Simulatio-

nen werden als eine Black-Box betrachtet, sodass diese Simulationsart für die Kopplung von grundsätzlich unterschiedlichsten Simulationen geeignet ist. Dabei muss jedoch die Ausgabe der vorläufigen Simulation als Eingabe der nachfolgenden Simulation genutzt werden können und dafür geeignet sein. Der Datenaustausch zwischen den Simulationen erfolgt zu diskreten Zeitschritten, die unabhängig von den Iterationszeitschritten der einzelnen Simulationen sein können [49, 32, 54]. Der Datenaustausch kann dabei über einen gemeinsamen Datenspeicher (engl.: *Shared Memory*) oder durch eine direkte Prozesskommunikation (engl.: *Inter-Process Communication*) erfolgen. Ein entscheidender Vorteil von gemeinsamen Datenspeichern ist, dass es weniger von den Funktionen und dem Zeitverhalten des Betriebssystems abhängt, auf die Solver ausgeführt werden und dadurch bessere Plattformunabhängigkeit sichergestellt werden kann [102]. Die einzelnen Simulationseinheiten können auf unterschiedliche Rechenressourcen und Plattformen verteilt werden und somit eine *verteilte* Co-Simulation erzeugen. Die parallele Ausführung der einzelnen Simulationseinheiten resultiert zudem in einer *parallelen* Co-Simulation [50].

Synchronisation und Interpolation

Eine verteilte und parallele Simulation bringt einige Herausforderungen mit sich, die bereits seit den 1970er Jahre erforscht werden [50]. Zu diesen Herausforderungen gehört das *Zeitmanagement* und speziell das *Synchronisationsproblem*, das infolge unterschiedlicher Simulationsgeschwindigkeiten[1] der einzelnen Teilnehmer auftritt.

Verteilte Co-Simulationen müssen synchronisiert werden, um eine zeitlich richtige Simulation der einzelnen Elemente zu gewährleisten. Die Funktion des Zeitmanagements wird von einem Orchestrator übernommen, der die simulierte Zeit steuert und Daten von Ausgaben zu Eingaben gemäß dem Simulationsszenario überführt [54]. Als Grundlage der Synchronisation wird ein Zeitstempel verwendet, welcher mit jedem Datenelement übertragen und zur zeitlichen Einordnung genutzt wird. Es wurden bereits unterschiedliche Konzepte und Algorithmen entwickelt, um dem Synchronisationsproblem zu begegnen. Diese Zeitmanagementkonzepte können in *konservative* und *optimistische Algorithmen* eingeteilt werden [50]. Bei konservativen Algorithmen nimmt die Sicherstellung der Berücksichtigung aller Zwischenschritte, eine zentrale Rolle ein. Wenn eine Simulation ein Zeitschritt verarbeitet hat und Daten für den nächsten Zeitschritt erhält, muss sichergestellt werden, dass keine weiteren Daten ankommen, die einen früheren Zeitschritt betreffen

[1] In einer Simulation muss zwischen zwei Zeitbegriffen unterschieden werden: der *simulierten Zeit* (auch *Modellzeit* genannt) t und der *Simulationszeit* τ. Dabei ist die Simulationszeit τ die Zeit, die auf der Wanduhr vergeht, bis eine Simulation durchgeführt wurde. Die simulierte Zeit dagegen ist der Zeitabschnitt, welcher in der Simulation abgebildet wurde [54]. So kann beispielsweise eine Stunde Simulationszeit vergehen, um eine Sekunde simulierte Zeit in einer umfangreichen Simulation abzubilden.

[50]. Bei diesen Verfahren muss beachtet werden, dass ein sogenanntes *Deadlock* (deu.: Blockade) auftreten kann. Bei einem Deadlock erwarten die Teilnehmer eine Nachricht, bevor sie mit der Bearbeitung fortfahren und die Simulation bleibt stehen. Eine Methode, um einen Deadlocks zu vermeiden, ist beispielsweise das Senden von *Null-Nachrichten* (engl.: *Null message*), die einen Zeitstempel ohne weitere Eingabedaten enthalten [50]. Konservative Synchronisation bringt zeitliche Nachteile mit sich, da die Sicherstellung, dass sich Datensätze nicht überholen können, die Simulation verlangsamt. Um die Nachteile zu beheben, wurden optimistische Ansätze entwickelt. Ein optimistischer Algorithmus kann Verstöße in der zeitlichen Reihenfolge erkennen und bietet Mechanismen, um sich von diesen zu erholen. Dadurch kann eine höhere Parallelisierung erreicht werden. Ein optimales Verfahren muss jeweils anwendungsfallabhängig ausgewählt werden muss [50].

Diskrete und kontinuierliche Simulation

Die Zustandsübergänge eines Systems und der dazugehörigen Simulation können entweder (annähernd) *diskret* oder *kontinuierlich* sein. Dies führt zu der Unterscheidung zwischen ereignisorientierter bzw. diskreter (engl.: *Discrete Event - DE*) und kontinuierlicher (engl.: *Continuous Time - CT*) Simulation, wobei auch hybride Formen möglich sind [54]. In einer ereignisorientierten Simulation erfolgt eine zeitlich sofortige Reaktion (engl.: *reactivity*) auf einen externen Stimulus. Das System kann seinen Zustand ohne Zwischenzustände ändern. Außerdem können in einem Zeitschritt mehrere Ereignisse eintreten, die alle verarbeitet werden, bevor die simulierte Zeit fortgesetzt wird (engl.: *transiency*). Um die zeitgleichen Ereignisse zu strukturieren wurde das Konzept der *superdichten Zeit* (engl.: *superdense time*) eingeführt, in dem zu jedem Zeitpunkt t ein Index n angegeben wird, um zeitgleich Ereignisse zu differenzieren [32]. In einer kontinuierlichen Simulation verändert sich der Zustand des Systems über die Zeit kontinuierlich. Da in einer Co-Simulation die Daten zwischen einzelnen Simulationseinheiten nicht beliebig schnell ausgetauscht werden, erfolgt dieser zu festgelegten Zeitpunkten, die als *Makrozeitschritte* bezeichnet werden [54].

Es existieren Anwendungen, in den die beiden Simulationsarten ereignisorientiert und kontinuierlich im Rahmen einer Co-Simulation kombiniert werden sollen. Insbesondere eingebettete Systeme sind in diesem Kontext bedeutsam, da hier heterogene Komponenten zusammengebracht werden und in hybriden Systemen resultieren. Während digitale Controller als diskrete Systeme abgebildet werden können, sind die mechanischen Komponenten durch kontinuierliche Simulationen repräsentiert [36]. Dabei besteht die Herausforderung, die Orchestrierung der unterschiedlichen Verhaltensweisen miteinander zu verbinden. Während bei kontinuierlicher Simulation sich der Zustand des Systems kontinuierlich ändert und die Ausgaben beispielsweise durch Interpolation und Extrapo-

lation angenähert werden können, trifft dies bei einer ereignisorientierten Simulation nicht zu. Ereignisorientierte Simulationen können zu einem Zeitschritt mehrere Zustände annehmen und es existiert oft kein Ansatz, die Ausgaben der Simulation zu inter- oder extrapolieren, da Sprünge auftreten können und somit die *Libschitzstetigkeit* nicht gegeben ist [54]. In diesen sogenannten *hybriden Systemen* trifft somit kontinuierliche Dynamik auf diskrete Zustandsänderungen und Ereignisse [32, 36]. Damit können die unterschiedlichen Simulationen nicht ohne Weiteres miteinander gekoppelt werden, da gänzlich unterschiedliche Annahmen bezüglich der Art der Ein- und Ausgaben vorliegen. In einer kontinuierlichen Simulation wird beispielsweise eine kontinuierliche Eingabe erwartet, während eine vorgeschaltete ereignisorientierte Simulation diskrete Ausgaben generiert [54]. Die hybriden Konzepte beschreiben zwei Herangehensweisen: die *hybride ereignisorientierte Simulation* und die *hybride kontinuierliche Simulation*. Die Lösungsansätze bestehen jeweils darin, um die Simulationseinheit einen sogenannten Wrapper (deu.: *Hülle*) zu konstruieren, der die Ein- und Ausgaben jeweils passend umwandelt. So werden beispielsweise für eine ereignisorientierte Simulation durch einen Wrapper die kontinuierlichen Eingaben in diskrete Werte umgewandelt, indem bei jeder Überschreitung eines Grenzwertes ein Ereignis generiert wird [54]. Die Wahl eines geeigneten Ansatzes hängt stark vom jeweiligen Anwendungsfall, von den Eigenschaften der betrachteten Simulationseinheit und der Konstellation der Simulationen ab. Wesentliche Herausforderungen bei der Generierung von hybriden Co-Simulationen sind nach einer Zusammenfassung von Gomes et al. *Semantische Adaption*, *prädiktive Schrittweite*, *zeitliche Ereignislokalisierung*, *Identifikation von Diskontinuität*, *Handhabung von Diskontinuität*, *Algebraische Schleifen*, *Zulässigkeit*, *Zeno Verhalten*, *Stabilität* und *Fehlerfortpflanzung* [54]. Veröffentlichungen zu den einzelnen Herausforderungen einer hybriden Co-Simulation können [54] entnommen werden. Broman et al. beschreiben in [32] die Anforderungen, die von einem Standard für hybride Co-Simulationen berücksichtigt werden sollten.

Für die Modellierung und Durchführung von hybriden Co-Simulationen wurden bereits Sprachen und Frameworks wissenschaftlich und praktisch erarbeitet. Übersichten der Modellierungssprachen und Werkzeuge für hybrides Systemdesign und -simulation bieten [36, 110, 54].

Neben eigenen, proprietären Entwicklungen können beispielsweise CAx-Werkzeuge herangezogen werden die auf Co-Simulation und die ganzheitliche Abbildung mechatronischer und Cyber-Physischer Systeme spezialisiert sind. Darunter sind *MATLAB/Simulink* [98] und *Modelica* zu nennen. Unter Modelica wird dabei die Modellierungssprache [104] und die Simulationsumgebung [105] verstanden, die sowohl als Open Source genutzt werden kann, als auch in einigen kommerziellen Simulationsumgebungen als Grundlage eingesetzt wird.

2.4.2 Architektur- und Schnittstellenstandards

In der Simulationsdisziplin existieren unterschiedliche Ansätze und Standards für die Architektur und Schnittstellen von Simulationen. Diese sind darauf ausgerichtet, die Interoperabilität zwischen unterschiedlichen Simulationswerkzeugen und Domänen, welche mittels der Simulationen beschrieben werden, sicherzustellen. Im Folgenden werden einige Ansätze auf diesem Gebiet vorgestellt und einige für diese Dissertation bedeutsame Konzepte genauer erläutert.

Eine frühe Architektur für Co-Simulationen bildet die *High Level Architecture* (HLA). Die vom US *Defense Modeling and Simulation Office* (DMSO) zunächst für den militärischen Kontext entwickelte Technologie kann verwendet werden, um eine verteilte Co-Simulation aus unabhängigen Simulationen und echten Systemen zu generieren. Dabei wird eine zentrale Einheit, genannt *Run-Time-Infrastructure* (RTI), eingesetzt, um die Einzelsimulationen zu koordinieren und zu synchronisieren. Die entwickelte Architektur ist durch die IEEE1516 normiert [67].

Eine generische Architektur für verteilte Systeme wurde von der *Object Management Group* (OMG) unter der Spezifikation *Common Object Request Broker Architecture* (CORBA) entwickelt [113]. CORBA ist eine objektorientierte Middleware, welche plattformunabhängige Protokolle und Dienste definiert, welche insbesondere bei verteilten und heterogenen Anwendungen zum Einsatz kommen. Als Vermittler wird ein sogenannter *Object Request Broker* (ORB) eingesetzt. Des Weiteren stützt sich CORBA auf die *Interface Definition Language* (IDL) zur Spezifikation der Schnittstelle eines Objekts, über jene auf die Daten und Methoden des jeweiligen Objekts zugegriffen werden kann [113].

Im Rahmen des Projekts ACOSAR (*Advanced Co-Simulation Open System ARchitecture*) wurde ein *Distributed Co-Simulation Protocol* (DCP) entwickelt, welches in der Lage sein soll, verteilte, sowohl echtzeitfähige, als auch nicht-echtzeitfähige Simulationen zu koppeln [79]. Das DCP unterstützt die Definition, Konfiguration und Ausführung von unterschiedlichen Simulationen und Tests, indem eine Master-Slave Architektur eingesetzt wird. An der verteilten Simulation können beliebige Systeme teilnehmen, womit auch physischen Komponenten integriert werden können. Voraussetzung dabei ist eine XML basierte Beschreibung der einzelnen teilnehmenden Slave-Elemente. In der XML Datei werden die Eingaben, Ausgaben, Parameter und Fähigkeiten der einzelnen Elemente definiert [56]. Das DCP wurde als ein *Modelica Association Project* (MAP) durch die Modelica Association standardisiert [103].

Die *Open-Service-Gateway Initiative* (OSGi) Allianz [117] entwickelte ein Java basiertes aber hardware- und herstellerunabhängiges Framework zur dynamischen Kopplung von Softwareelementen zur Laufzeit. Die einzelnen Komponenten in einem OSGi basiertem Netzwerk sind gekapselt und interagieren über Services miteinander [99]. Oppelt et

AL. [116] nutzten beispielsweise OSGi, um eine Co-Simulation zu erzeugen, in der die einzelnen Simulationen über Simulations-Koppler angebunden sind und damit auch der Datenaustausch und Synchronisation ermöglicht wird [74].

Ein weiterer Ansatz der Co-Simulation basiert auf *Agenten-Architekturen*. Agenten stellen kooperierende verteilte Systeme dar und sind für verteilte Simulationen geeignet. Ein Agent ist dabei eine abgrenzbare Hardware und/oder Software-Einheit mit definierten Zwecken und Zielen, welche der Agent durch selbstständiges Verhalten versucht zu erreichen [20]. Die einzelnen Agenten agieren in der Regel in einem *Agenten-Netzwerk*. Die Agenten tauschen in dem Agenten-Netzwerk Nachrichten aus und können somit ein übergeordnetes Ziel verfolgen [20]. Im Kontext einer verteilten, agenten-basierten Simulation werden die einzelnen Simulationen über spezielle Schnittstellen an die Agenten angebunden und somit in das Agenten-Netzwerk integriert [74]. Agenten-basierte Konzepte für verteilte Simulationen und für den Digitalen Zwilling bringen ebenfalls einige Herausforderungen mit sich. Es müssen die oben beschriebenen Anforderungen an eine Co-Simulation, wie beispielsweise ein Synchronisationsmechanismus, erfüllt werden. Agenten-Systeme werden im Kontext von Co-Simulationen und des Digitalen Zwillings erforscht, da diese die Möglichkeit bieten, die Konstellation der Gesamtsimulation zur Laufzeit zu verändern, indem neue Agenten und damit auch Simulationen hinzukommen und wieder verschwinden können. JUNG ET AL. entwickelten ein domänenunabhängiges Multi-Agenten System für Co-Simulation, welches die dynamische Kopplung von Simulationen auch aus unterschiedlichen Simulationswerkzeugen ermöglicht. Die einzelnen Simulations-Agenten können zur Laufzeit eingebunden und wieder entfernt werden [74]

Weitere Informationen zu den unterschiedlichen Architekturen und Technologien für Co-Simulationen und verteilte Simulationen können [56, 73, 56, 79, 50] entnommen werden. Eine Übersicht nach den jeweiligen Domänen bietet [54]. Außerdem existieren domänenspezifische Co-Simulation Ansätze [73, 74, 56, 54], wie z.B. das *Open Simulation Interface* (OSI) [97], das *System Structure and Parametrization* (SSP) [106], die *Distributed Interactive Simulation* (DIS) [66] und AUTOSAR (*AUTomotive Open System ARchitecture*) [51], die im Rahmen der Dissertation jedoch nicht weiter erläutert werden.

Functional Mockup Interface (2P)

Das *Functional Mockup Interface* (FMI) ist ein simulationswerkzeugunabhängiger Standard für den Austausch von Simulationsmodelle und für Co-Simulation [26, 107]. Die Entwicklung von FMI hat seinen Ursprung in dem ITEA2 (*Information Technology for European Advancement*) Projekt MODELISAR und wurde von der Daimler AG initiiert [27]. Der FMI Standard zielt auf zwei unterschiedliche Szenarien ab - dem Austausch von Simulationsmodellen (*model exchange*) und die Co-Simulation.

Der Grundgedanke bei dem Austausch von Simulationsmodellen ist die Bereitstellung von dynamischen Systemmodellen in Form eines werkzeugneutralen Simulationselements. Dieses in der Programmiersprache C geschriebene Simulationselement wird aus einer proprietären Simulationsumgebung exportiert, hat definierte Ein- und Ausgabevariablen und kann mittels unterschiedlicher Simulationswerkzeuge simuliert werden [27, 107].

Die Co-Simulation des FMI dient dazu, mehrere, aus unterschiedlichen Simulationswerkzeugen stammende, Simulationsmodelle miteinander zu koppeln und diese in einer gemeinsamen Co-Simulationsumgebung auszuführen [27, 107]. Die einzelnen Simulationssubsysteme werden bei der Co-Simulation in den einzelnen Simulationswerkzeugen ausgeführt, wobei die Kommunikation auf diskrete Kommunikationszeitpunkte beschränkt ist, die variabel gestaltet werden können. In der Zeit zwischen den einzelnen Kommunikationsereignissen werden die Subsysteme durch die individuelle Solver gelöst. Der Datenaustausch zwischen den Subsystemen sowie deren Synchronisation wird durch einen sogenannten *Master*-Algorithmus gesteuert, welche die einzelnen *Slave*-Simulationen verwaltet [26]. Der Master-Algorithmus ist somit der Orchestrator im Rahmen von FMI.

Ein Simulationselement, welches nach dem FMI-Standard implementiert ist, wird als eine *Functional Mockup Unit* (FMU) bezeichnet und steht in Form eines ZIP Containers mit der Dateiendung *.fmu* zur Verfügung [26]. Der FMU-Container beinhaltet eine XML Datei, Programmfunktionen in der C-Sprache und weiteren Daten. Die XML-Datei definiert und beschreibt die für die Simulation benötigten Variablen. Die C-Funktionen beinhalten im Falle vom Austausch von Simulationsmodellen die benötigten Modellgleichungen zur Berechnung der Simulation. Im Anwendungsfall der Co-Simulation kann mittels der C-Funktionen die Kommunikation zwischen den Simulationswerkzeugen initiiert, ein Kommunikationszeitschritt ausgeführt und die Daten an den Kommunikationspunkten ausgetauscht werden [107]. Weitere Daten können Dokumentation, Diagramme und Tabellen beinhalten, welche die Modelle und Simulationen erfordern sowie Programmbibliotheken, die bei der Ausführung benötigt werden [27].

Eine weitere Einsatzmöglichkeit von FMI ist die Integration von FMUs in eingebettete Elektronik, um Simulationsmodelle direkt auf eingebetteter Elektronik bereitstellen zu können [3, 114]. Mit diesem Ansatz können beispielsweise physikalischen Modelle in die *Electronic Control Units* (ECUs) von Fahrzeugen integriert werden, um die ECUs mit erweiterten Funktionen, wie beispielsweise modellbasierter Vorhersage ausstatten zu können [114]. Die FMUs for Embedded Systems (eFMUs) unterliegen durch den Einsatz auf ECUs besonderen Anforderungen. Die ECU Software muss strikten Entwicklungsrichtlinien (z.B. gemäß MISRA - *Motor Industry Software Reliability Association*) genügen und für spezialisierte Hardware- und Softwarearchitekturen (z.B. AUTOSAR) ausgelegt sein [3]. Die Entwicklung von eFMI mündet in einem FMI erweiternden Standard [47].

Die Voraussetzung für den Einsatz von FMI ist, dass die beteiligten Domänen Werkzeuge

besitzen, die den FMI Standard ausreichend unterstützen, um eine ganzheitliche Co-Simulation aufbauen zu können [23]. Die FMI Technologie in der aktuellen Version eignet sich jedoch aus einem anderen Grund nur bedingt für die Co-Simulation im Kontext des Digitalen Zwillings. Vor dem Start der Simulation müssen alle Teilnehmer bekannt sein und bei der Initiierung bereits Informationen an den Master-Algorithmus liefern [17]. Deshalb ist FMI für die dynamische Co-Simulation Digitaler Zwillinge wenig geeignet, da die Simulationselemente zur Laufzeit hinzugefügt und entfernt werden können müssen [74]. Eine weitere Erschwernis ist die fehlende, explizite Unterstützung von hybriden Co-Simulationen in der FMI Version 2.0. [32].

OPC UA

Die *Open Platform Communications Unified Architecture* (OPC UA) ist eine hersteller- und plattformunabhängige, serviceorientierte Kommunikationsarchitektur. Die von der *OPC Foundation* herausgegebene und weiterentwickelte Spezifikation ist in der Normenreihe DIN EN IEC 62541 [44], die sich zum Zeitpunkt der Erstellung der Dissertation teilweise im Entwurfsstadium befindet, definiert. OPC UA verbindet die Anforderungen der Industrie 4.0, der Maschine-zu-Maschine Kommunikation und des Internets der Dinge und stellt sowohl die Kommunikationsmechanismen als auch die Grundlagen der Informationsmodellierung bereit [115]. Die Interoperabilität als Kerngedanke von OPC UA ermöglicht sowohl die horizontale Kommunikation zwischen Systemen auf einer Ebene der Automatisierungspyramide, als auch die Kommunikation zwischen unterschiedlichen Ebenen in der vertikalen Sichtweise [96].

Die beiden Grundsäulen von OPC UA sind einerseits der *Datentransport* und andererseits die OPC UA *Informationsmodellierung*. Beim Datentransport werden mehrere Technologien unterstützt. Die Daten können einerseits mithilfe von TCP/IP[2]-Protokollen übertragen werden, was insbesondere für die Intranet-Kommunikation geeignet ist [88]. Andererseits können die Daten über das Internet mittels HTTP und XML basierter Kommunikation übertragen werden [96]. Als Codierung kann dabei entweder ein XML-Stream (*UA XML*) oder eine binäre Form (*UA Binary*) eingesetzt werden. Die Sicherung der Daten wird während des Datentransports durch Sicherheitsmechanismen auf Basis eines *Security Models* bewerkstelligt [88].

Ein OPC UA Informationsmetamodell als zweite Grundsäule definiert Regeln zur Erzeugung von Informationsmodellen und gibt Basiskonstrukte zur Modellierung vor [96]. Die beiden Grundsäulen formen die Basis für weitere OPC UA Services, welche die Kommunikation zwischen einem OPC UA Server und einem OPC UA Client definieren. Für

[2]TCP - Transmission Control Protocol
IP - Internet Protocol

die Informationsmodellierung werden OPC UA Basisinformationsmodelle bereitgestellt, die sich selbst definierende Informationen enthalten und damit die Grundlage für die Kommunikation zwischen Maschinen sicherstellen. Mit der dadurch definierten Semantik und Syntax können Daten von Maschinen selbständig interpretiert werden [88]. Das OPC UA Informationsmodell stellt einen Adressraum bereit, in dem *Knoten* und *Referenzen* definiert sind. Die Knoten werden in acht Knotenklassen unterteilt und mittels definierter Referenzen zu selbst-beschreibenden Netzwerken kombiniert [96].

Das Basisinformationsmodell wird durch spezifische Informationsmodelle für einzelne Systemklassen aus unterschiedlichen Branchen, den sogenannten *Companion Specifications* (CS), erweitert. Die Companion Specifications werden durch Standardisierungsgremien erarbeitet und bilden die Grundlagen für spezialisierte technologische Anwendungen [115]. Des Weiteren besteht die Möglichkeiten, auf dieser umfangreichen Basis zusätzliche, eigene Informationsmodelle zu definieren und die bestehenden nach eigenen Anforderungen anzupassen und zu erweitern [88]. Die benutzerdefinierten Änderungen können jedoch ein Hindernis bei nativer Kompatibilität sein und müssen unter den Kommunikationspartnern zusätzlich verständigt werden.

Durch die weitgehende Standardisierung von OPC UA, durch die Unterstützung unterschiedlicher Datentransportmechanismen, der Verwendung von Informationsmodellen zur Abbildung der Semantik der Daten und durch ein integriertes Sicherheitskonzept ist es für Industrie 4.0 Anwendungen geeignet [157]. Zu diesen Anwendungsfällen gehört auch die Verknüpfung eines physischen Systems mit dessen Digitalen Zwilling [86]. OPC UA kann jedoch auch genutzt werden, um den Datenaustausch innerhalb von Co-Simulationen zu verwalten. Hensel et al. entwickelten beispielsweise eine Co-Simulation unter Verwendung von OPC UA, in der mittels individueller Adapter und eines OPC UA Clients/Servers die Simulationen mittels eines zentralen OPC UA Servers miteinander verbunden wurden und über ein Master/Slave System koordiniert wurden [64].

2.4.3 Virtuelle Plattformen

Die Hardware eingebetteter Systeme wird, der Vorhersage durch das *Moor´sche Gesetz* folgend, immer leistungsfähiger. Dementsprechend steigt der potenzielle Funktionsumfang der zugehörigen eingebetteten Software, wobei gleichzeitig immer höhere Qualitäts-, Kosten- und Zeitanforderungen gestellt werden [13, 111]. Die rechtzeitige Auslieferung von eingebetteten Systemen ist von entscheidender Bedeutung, weil nicht nur der Markt von einem starken Wettbewerb getrieben ist, sondern auch die Kundenwünsche sich schnell verändern [146, 38]. Eine Maßnahme zur Begegnung dieser Herausforderungen ist die Verwendung von *Virtuellen Plattformen Eingebetteter Systeme*, die es ermöglichen, bei der Entwicklung eingebetteter Systeme auf die Zielhardware (engl.: *Target Hardware*)

und physische Entwicklungselektronik zu verzichten und die Entwicklung vollständig virtuell auf einem Entwicklungscomputer (*Host Hardware*) stattfinden zu lassen. Dabei reduzieren Virtuellen Plattformen die Zeit und den Aufwand zur Entwicklung eingebetteter Anwendungen [13]. Die Zeitersparnis resultiert daraus, dass die Softwareentwicklung bereits beginnen kann, bevor die Hardware oder ein physischer Prototyp des Systems existiert [38]. Virtuelle Plattformen werden nicht nur eingesetzt, um eingebettete Software ohne zugehörige Hardware zu entwickeln, sondern auch zur Verifikation der noch in Entwicklung befindlichen Hardware [146]. Dies verkürzt zusätzlich den Gesamtentwicklungsprozess, da hier Probleme bereits früh erkannt und beseitigt werden können. Weiterhin unterstützt die zeitliche Überlagerung der Hard- und Softwareentwicklung das Co-Design von Hard- und Software. Virtuelle Plattformen, welche die Hardwaresimulation mit Softwareentwicklungswerkzeugen vereinen, werden auch *Virtual Development Environment* (VDE) genannt.

Eine Klasse der Virtuellen Plattformen basiert auf einer sogenannten *Host-Simulation*. Bei der Host-Simulation wird der Code der eingebetteten Software für die Architektur des Host-Prozessors kompiliert und direkt auf der Host-Hardware ausgeführt. Diese Klasse der Simulation benötigt die Simulationsmodelle der Peripherieelemente der Zielhardware, um diese bei der Ausführung der eingebetteten Software zur Verfügung stellen zu können [13].

Die Abbildung 2.6 zeigt in Anlehnung an [13] einige Soft- und Hardwareelemente eines typischen eingebetteten Systems. Das System besitzt beispielsweise als Mensch-Maschine-Schnittstelle einen LCD (*Liquid Crystal Display*) Bildschirm und Eingabetasten. Die Anwendungssoftware wertet die Eingaben des Benutzers aus und zeigt mittels des Bildschirms Ausgaben an. Zur Verarbeitung der Eingaben benötigt die Anwendungssoftware Funktionen der Middleware sowie des Echtzeit-Betriebssystems, die mittels einer API bereitgestellt werden. So werden beispielsweise grundlegende Funktionen, wie die Darstellung der Grafik auf einem Bildschirm oder das Management von Eingaben nicht in der Anwendungssoftware implementiert, sondern durch die Middleware bereitgestellt. Das Echtzeitbetriebssystem (engl.: *Real Time Operating System (RTOS)*) beinhaltet zudem zielhardwarespezifische Komponenten. Die Middleware und das Betriebssystem können nur betrieben werden, wenn die Zielhardware tatsächlich vorhanden ist. Die Abbildung 2.6 zeigt damit auch, dass die Anwendungssoftware zwar nicht direkt von der Zielhardware abhängt, jedoch über die Middleware und die Treiber sowie das Betriebssystem einen indirekten Zusammenhang herstellen [13]. Bei einer Host-Simulation werden die Anwendungssoftware und die Middleware mittels eines *Hostcompilers* für das Hostsystem kompiliert und nativ auf dem Hostsystem ausgeführt. Die Gerätetreiber werden durch simulierte Gerätetreiber ersetzt, welche wiederum mit den Peripheriegeräte-Simulationsmodellen verknüpft sind. Das Echtzeitbetriebssystem wird ebenfalls durch ein

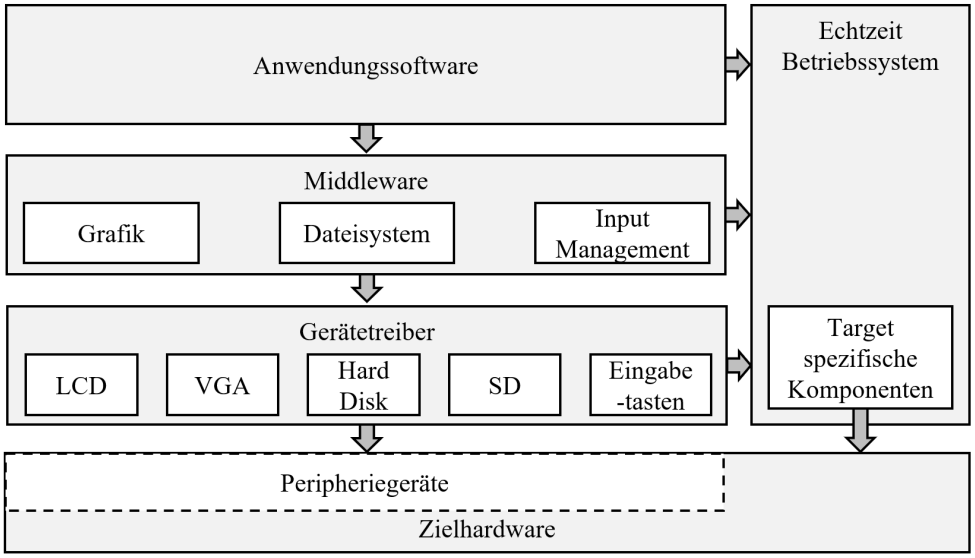

Abbildung 2.6: Soft- und Hardwareelemente eines typischen eingebetteten Systems [In Anlehnung an [13]]

Simulationsmodell ersetzt [13].

Eine weitere Klasse von Virtuellen Plattformen basiert auf einem sogenannten *Instruction Set Simulator* (ISS). Ein Beispiel für ISS ist *Open Virtual Platforms* (OVP) [119]. Ein ISS überführt ein Programm, welches für ein Zielsystem entwickelt und kompiliert wurde mittels einer binären Übersetzung zur Laufzeit in Anweisungen des Hostsystems. Dies erlaubt das Testen von Programmen, ohne diese für die Plattform kompilieren zu müssen, auf welcher es tatsächlich ausgeführt wird [146].

Die beiden vorgestellten Klassen von Virtuellen Plattformen bringen jeweils Vor- und Nachteile mit sich, weshalb in einem konkreten Anwendungsfall die Eignung überprüft werden muss. So bringt die hohe Simulationsgeschwindigkeit von ISS jedoch auch den Nachteil, dass die Zeit in der Ausführung nicht berücksichtigt wird [146].

In der Domäne eingebetteter Systeme existieren unterschiedliche Plattformen und Werkzeuge zur Entwicklung und Simulation der eingebetteten Hardware und Software. Neben offenen Systemen wie *Open Virtual Platforms* [119], werden diverse kommerzielle Systeme angeboten. Zu diesen gehören beispielsweise Produkte der Firmen Synopsys [147], ARM [14], Siemens [137], Labcenter Electronics [87] und weitere. Eine Übersicht einiger Werkzeuge, die zu sogenannten *Electronic Design Automation* (EDA) Tools zugeordnet werden können, bietet [95].

2.5 Fazit zum Stand der Technik

Der in diesem Kapitel dargestellte Stand der Technik beschreibt die notwendigen Grundlagen zum Verständnis des Konzepts einer Virtuellen Klemmleiste als eine virtuelle Software-Hardware Schnittstelle zur Integration der Verhaltenslogik in den Digitalen Zwilling. Die Ausgangsbasis bildet dabei die Virtuelle Produktentwicklung technischer Systeme (Kapitel 2.1) und das Industrie 4.0-Konzept 2.2. Der Produktentstehungsprozess unterscheidet in den einzelnen Disziplinen der Produktentwicklung technischer Systeme, wobei auch Ähnlichkeiten in den Entwicklungsmethoden vorhanden sind. Eine disziplinübergreifende Entwicklungsmethodik bietet das V-Modell nach VDI/VDE 2206 [156], indem es die Disziplinen der Mechanik, Elektrotechnik und Softwaretechnik bündelt, um der Besonderheit Cyber-Physischer Systeme hinsichtlich der Flexibilität und Wandlungsfähigkeit gerecht zu werden. Das V-Modell unterstützt die Entwicklung physischer Systeme und kann genutzt werden, um die zugehörigen Digitalen Zwillinge zu entwickeln. Die Entwicklung eines Digitalen Zwillings unterscheidet sich jedoch hinsichtlich der Zusammenarbeit der einzelnen Disziplinen, indem die Entwickler auf neuen Gebieten miteinander kooperieren müssen. Während beispielsweise die Modelle und Werkzeuge der Entwicklung von eingebetteter Elektronik bisher noch weitgehend innerhalb der Domäne verblieben, müssen diese für die Integration des Verhaltens in den Digitalen Zwilling extern bereitgestellt werden können. Die Verwendung der Elektronikmodelle und der zugehörigen Werkzeuge im Kontext des Digitalen Zwillings erfordert wiederum die Einführung von neuen Konzepten und Schnittstellen sowie die Schaffung einer gemeinsamen Semantik. Die Entwickler der mechanischen Komponenten stehen ebenso vor neuen Herausforderungen. Insbesondere müssten die in der Entwicklung erstellten und verwendeten Simulationen für den Digitalen Zwilling weitergedacht werden. Die Simulationsmodelle und -werkzeuge unterliegen im Digitalen Zwilling neuen Anforderungen, da dieser in der Nutzungsphase des Produkts existiert und jede Produktinstanz von Simulationen begleitet wird. Diese neuen Herausforderungen, Anforderungen, Konzepte und Schnittstellen müssen noch wissenschaftlich erarbeitet werden. Die Forschung zu Industrie 4.0 bietet bereits Konzepte und Technologien, welche für die Weiterentwicklung zum Digitalen Zwillings herangezogen werden können. Dazu gehört z.B. die Verwaltungsschale, welche als Rahmenwerk des Digitalen Zwillings eingesetzt werden kann oder die OPC UA Technologie, dessen semantischen Informationsmodelle und Kommunikationsschittstellen verwertet werden können.

Im Kapitel 2.3 wird das Konzept des Digitalen Zwillings beschrieben. Die vorgestellten Definitionen des Digitalen Zwillings zeigen auf, dass das Verhalten des korrespondierenden Physischen Zwillings ebenfalls repräsentiert werden muss. Dies wird in den ausgearbeiteten Charakteristika des Digitalen Zwillings ebenfalls adressiert. Grieves und Vickers

stellten Verhaltenskategorien auf und entwickelten einen Test (*GTV*) zur Beurteilung der Realitätsnähe [59]. Die existierenden Ansätze des Digitalen Zwillings nennen jedoch keine Lösung der ausformulierten Problemstellung, dass das Verhalten des Digitalen Zwillings sich möglichst dem Verhalten des physischen Systems auf Basis gleicher Verhaltenslogik nähern soll. Die beschriebenen Architekturen und Schnittstellen des Digitalen Zwillings schließen diese Möglichkeit zwar nicht aus - jedoch zeigen diese keine konkrete Lösung auf.

Das Kapitel 2.4 adressiert die Simulation im Digitalen Zwilling und geht insbesondere auf die Co-Simulation ein. Die Methoden der Co-Simulation ermöglichen die Kopplung von mehreren Simulationen und schaffen die Grundlage für eine Simulationskette im Digitalen Zwilling. Diese kann gemäß den vorgestellten Architekturen und Schnittstellenstandards implementiert werden. Es zeigt sich jedoch auch, dass die beabsichtigte Kopplung einer Elektroniksimulation mit Partialmodellen des Digitalen Zwillings bisher nicht explizit beschrieben wurde. Stattdessen müssen geeignete Technologien wie FMI und OPC UA im Rahmen einer neuartigen Lösung instrumentalisiert werden.

Im Stand der Technik wurden somit die Voraussetzungen erarbeitet, welche für die anvisierte Weiterentwicklung des Digitalen Zwillings notwendig sind. Die Darstellung des aktuellen technischen und wissenschaftlichen Stands zeigt jedoch auch, dass obwohl einige technologische Voraussetzungen bereits gegeben sind, bisher noch kein disziplinübergreifendes Konzept existiert, um den Digitalen Zwilling eines eingebetteten Systems mit der gleichen Verhaltenslogik auszustatten, wie sie im Physischen Zwilling eingesetzt wird. Diese Erkenntnis führt zur Erarbeitung eines entsprechenden Konzepts in den nächsten Kapiteln.

3 Handlungsbedarf und Anforderungsprofil

Im nachfolgenden Kapitel wird der Handlungsbedarf zur Konzeption einer Software-Hardware Schnittstelle zur Integration der Verhaltenslogik in den Digitalen Zwilling hergeleitet. Der im Teilkapitel 3.1 identifizierte Handlungsbedarf resultiert aus der Analyse des Stands der Technik und der Forschungsarbeiten im Kontext des Digitalen Zwillings. Im darauffolgenden Teilkapitel 3.2 wird aus dem Handlungsbedarf eine Zieldefinition der vorliegenden Dissertation abgeleitet. Im Teilkapitel 3.3 werden anschließend Anwendungsfälle mittels eines Anwendungsfalldiagramms beschrieben, um die Systemgrenze, die relevanten Akteure und das Zusammenwirken der einzelnen Aktivitäten präzise zu spezifizieren. Abschließend werden im Teilkapitel 3.4 Anforderungen an eine Schnittstelle zur Integration der Verhaltenslogik in den Digitalen Zwilling definiert.

3.1 Ableitung des Handlungsbedarfs

Der Digitale Zwilling findet in immer mehr Branchen Anwendung und es wurden bereits mögliche Anwendungsfällen beschrieben. Jedoch bestehen weiterhin Hürden, die einen durchgehenden Einsatz über alle Produktlebensphasen und die Nutzung als Ergänzung und Erweiterung des korrespondierenden physischen Systems verhindern. Bisher werden die vom physischen System und aus anderen Quellen kommenden Daten genutzt, um beispielsweise Analysen, Vorhersagen und Visualisierungen mittels des Digitalen Zwillings zu ermöglichen [71]. Damit können bereits einige der im Stand der Technik beschriebenen Charakteristika und Anwendungsfälle des Digitalen Zwillings realisiert werden.

Physische Systeme mit eingebetteter Software, wie beispielsweise Cyber-Physische Systeme, können aktuell jedoch nicht ganzheitlich virtuell abgebildet werden, weil die Berücksichtigung der Verhaltenslogik des Physischen Zwillings fehlt. Unter Verhaltenslogik wird im Rahmen der Dissertation die eingebettete Software und eingebettete Elektronik und insbesondere deren dynamisches Zusammenwirken verstanden. Gemäß der Darstellung im Stand der Technik wird zum aktuellen Zeitpunkt weder die eingebettete Elektronik, noch die eingebettete Software im Digitalen Zwilling originalgetreu bzw. realitätsnah repräsentiert. Demnach unterscheidet sich der Digitale Zwilling eines Cyber-Physischen

Systems hinsichtlich der eingebetteten Datenverarbeitung maßgeblich von dem Physischen Zwilling. Dadurch können Datenverarbeitungsprozesse die innerhalb des physischen Produktes ablaufen nicht realitätsnah simuliert werden. Für diesen Einsatzzweck muss die Verhaltenslogik des physischen Systems im Digitalen Zwilling virtualisiert werden, um eine Äquivalenz der beiden Zwillinge zu erreichen. Während der Entwicklung von Elektronik werden bereits Simulationswerkzeuge eingesetzt, um die entsprechende Verhaltenslogik zu virtualisieren. Die Verwendung ist jedoch auf die Elektronikentwicklung beschränkt und es existieren keine konkreten Ansätze, diese Werkzeuge im Digitalen Zwilling einzusetzen. Dadurch existieren auch keine Technologien, um diese Werkzeuge in der disziplinübergreifenden Anwendung und in der Nutzungsphase des Produkts einzusetzen. Insbesondere werden Schnittstellen benötigt, um eine Elektroniksimulation mit mechanischen Partialmodellen des Digitalen Zwillings zu koppeln. Im Physischen Zwilling sind es die Schnittstellen zwischen der Software und Hardware des Systems. Der Handlungsbedarf besteht vereinfacht ausgedrückt darin, eine *virtuelle Software-Hardware Schnittstelle* einzuführen, um das Verhalten in den Digitalen Zwilling integrieren zu können.

Ein Hemmnis bei der Integration der Verhaltenslogik in den Digitalen Zwilling sind zudem die fehlenden unterstützenden Methoden zur Erstellung und Zusammenführung von Modellen und Werkzeugen. Dadurch ist ein solches Vorhaben stets ein individueller ergebnisoffener Prozess und dadurch riskant. Es ist nicht erforscht, welche Modelle für die Integration der Verhaltenslogik verwendet werden müssen, wie diese entstehen und welches Fachwissen benötigt wird, um einen sinnvollen Einsatz zu ermöglichen. Neben Neuentwicklung der Modelle eines Digitalen Zwillings ist beispielsweise die Wiederverwendung von Entwicklungsmodellen aus der interdisziplinären Produktentwicklung ein möglicher Ansatz zur Erweiterung eines Digitalen Zwillings durch die Verhaltenslogik. Doch auch im Entwicklungsprozess nach dem V-Modell, in dem die unterschiedlichen Disziplinen bereits Schnittstellen untereinander aufweisen, ist es nicht offensichtlich, wie die Zusammenarbeit konkret aussehen muss, um neben dem physischen Produkt auch dessen Digitalen Zwilling mit der gleichen Verhaltenslogik zu versehen. Dabei ist gerade die Produktentwicklung für die Erschaffung eines Digitalen Zwillings gut geeignet, da in dieser Phase die Disziplinen eng zusammenarbeiten und das Produktfachwissen vorhanden ist. Insbesondere können Modelle gemäß dem Konzept des Digitalen Masters [143] zur Erweiterung des Digitalen Zwillings verwendet werden. Dadurch wird der Nutzen der in der Produktentwicklung entstehenden Modelle erhöht. Aus dieser Begründung heraus wird der Handlungsbedarf dahingehend konkretisiert, dass die Integration der Verhaltenslogik bereits während der multi-disziplinären Produktentwicklung stattfindet soll, um die Master-Modelle wiederverwenden zu können. Dazu wird ein strukturiertes Vorgehen benötigt, wie die einzelnen Disziplinen zusammenwirken müssen, um die Integration der Verhaltenslogik in den Digitalen Zwilling durchzuführen.

Zusammenfassend besteht der Handlungsbedarf darin, Methoden und Werkzeuge zur Integration der Verhaltenslogik zu erarbeiten, welche die Modelle und das Fachwissen aus der Produktentwicklung technischer Systeme wiederverwendet. Diesen Handlungsbedarf adressiert die vorliegende Dissertation.

3.2 Zieldefinition

Aus dem beschriebenen Handlungsbedarf wird im Folgenden die Zieldefinition abgeleitet. Die Zielsetzung der Dissertation liegt in der Konzeption eines Ansatzes zur Integration der Verhaltenslogik in den Digitalen Zwilling. Um dieses Ziel zu erreichen, müssen einerseits Methoden beschrieben werden, diese Integration als einen multi-disziplinären Prozess durchzuführen. Andererseits werden Werkzeuge benötigt, welche die Integration ermöglichen und unterstützen.

Der gesetzte Fokus der Dissertation liegt auf multi-disziplinären Systemen, die durch das Zusammenwirken von Software, Elektrik/Elektronik und Mechanik gekennzeichnet sind. Die multi-disziplinäre Zusammenarbeit findet hauptsächlich während der Produktentwicklung statt. Aus diesem Grund muss die Beschreibung der methodischen Vorgehensweise in dieser Produktlebensphase ansetzen. Dabei sollen die bestehende Infrastruktur, die Prozesse, die Technologien und die Entwicklungsmethoden nicht grundsätzlich verändert werden müssen. Dadurch kann ein Digitaler Zwilling mit integrierter Verhaltenslogik als Ergänzung des physischen Produkts entstehen und für neue Geschäftsmodelle bereitgestellt werden.

Bei dem Prozess der Integration der Verhaltenslogik ist es dabei zielführend, in den einzelnen Disziplinen bereits etablierte Technologien zu verwenden, um einen Mehraufwand möglichst gering zu halten. Der vorgestellte Stand der Technik zeigt auf, dass die einzelnen Disziplinen während der Produktentwicklung bereits virtuelle Modelle einsetzen. Im Rahmen eines Ansatzes zur Integration der Verhaltenslogik in den Digitalen Zwilling muss nun die Verbindung zwischen den Modellen geschaffen werden.

Das konkretisierte Ziel der vorliegenden Dissertation ist somit die Entwicklung eines Ansatzes zur Integration der virtuellen Verhaltenslogik in den Digitalen Zwilling unter Berücksichtigung bereits existierender Methoden, Modelle und Werkzeuge der einzelnen Disziplinen.

3.3 Anwendungsfälle

Nach Ableitung des Handlungsbedarfs und Festlegung der Zieldefinition erfolgt die Betrachtung der im Rahmen dieser Dissertation adressierten Anwendungsfälle. Die Anwen-

dungsfälle bilden den Rahmen des zu entwickelnden Konzepts, indem sie die Systemgrenze definieren und die Aktivitäten aller beteiligter Akteure beinhalten. Die Anwendungsfälle werden in der UML (*Unified Modeling Language*) Notation visualisiert, welche mit dem Anwendungsfalldiagramm eine Möglichkeit bietet, in der frühen Systementwicklung das Zusammenwirken beteiligter Akteure zu beschreiben. Weilkiens definiert einen Anwendungsfall wie folgt:

„Ein Anwendungsfall (engl. use case) beschreibt eine Menge von Aktionen eines Systems, die zu einem beobachtbaren Ergebnis führen, das typischerweise für die Akteure oder Stakeholder einen Wert hat.“ [162].

Das UML Anwendungsfalldiagramm determiniert nicht die tatsächliche Implementierung, sondern bietet eine übersichtliche Darstellung der geplanten Funktionalität des Systems aus der externen Sicht des Nutzers [127]. Ein Anwendungsfalldiagramm begrenzt sich auf die Beschreibung der Interaktionen zwischen den Akteuren und gibt keine Auskunft über die interne Verarbeitung oder die Reihenfolge der Aktivitäten [154].

Die beiden im Folgenden dargestellten Anwendungsfälle beschreiben einerseits die Integration der Verhaltenslogik in den Digitalen Zwilling und andererseits die Steuerung des Digitalen Zwillings, die durch die vorige Integration ermöglicht wird. Beide Anwendungsfälle sind bedeutend, um darauf aufbauend die Anforderungen an das Konzept zur Integration der Verhaltenslogik in den Digitalen Zwilling auszuarbeiten.

3.3.1 Integration des Verhaltens in den Digitalen Zwilling

Zur Verdeutlichung des Zusammenwirkens der Akteure bei der Integration der Verhaltenslogik in den Digitalen Zwilling wird ein UML Anwendungsfalldiagramm verwendet. Die Systemgrenze bezieht sich dabei nicht auf ein konkretes System, sondern auf das übergeordnete Zusammenwirken aller Systeme, die zur Integration der Verhaltenslogik notwendig sind. Der Anwendungsfall ist in Abbildung 3.1 dargestellt. Das Diagramm beschreibt das Zusammenwirken der Akteure *Softwareingenieur*, *Elektronikingenieur*, *Systemingenieur*, *Kunde* und *Konstrukteur*.

Den Akteuren des Anwendungsfalldiagramms werden Aktionen zugeordnet, die zur Erreichung des beabsichtigten Ergebnisses, in Form der Integration der Verhaltenslogik in den Digitalen Zwilling, notwendig sind. Die Aktionen weisen Abhängigkeiten untereinander auf, da Aktionen auf die Durchführung anderer Aktionen angewiesen sind. Die einzelnen Aktionen des abgebildeten Anwendungsfalls können ebenfalls als Anwendungsfälle detailliert beschrieben werden und werden oft als solche bezeichnet. Zur deutlichen Trennung zwischen den übergeordneten und den untergeordneten Anwendungsfälle werden die letzteren gemäß der Definition von Weilkiens als *Aktionen* bezeichnet.

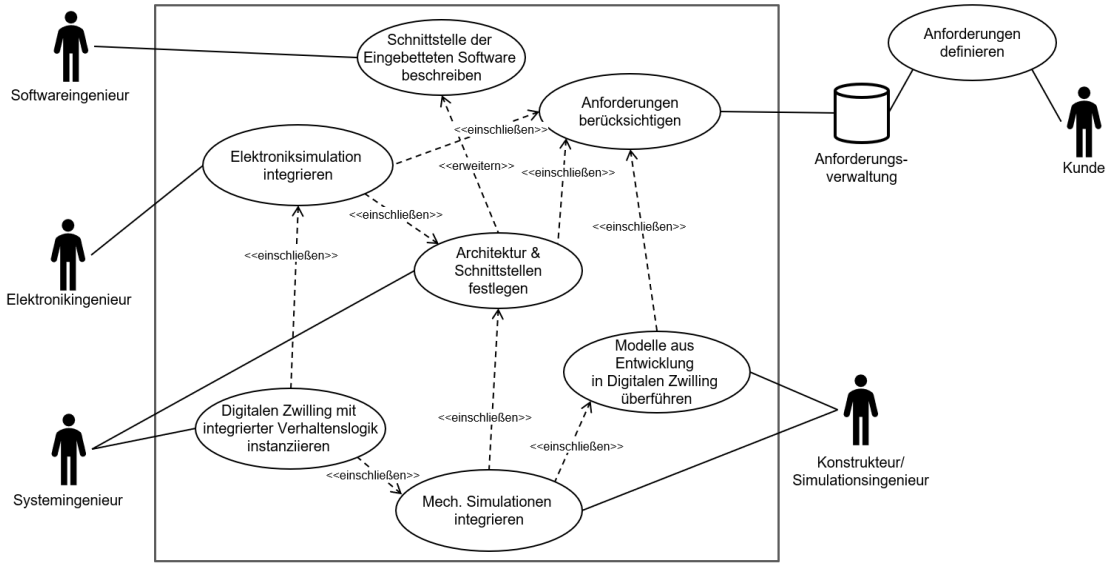

Abbildung 3.1: Anwendungsfalldiagramm zur Integration der Verhaltenslogik in einen Digitalen Zwilling

Systemgrenze

Die Systemgrenze umfasst die Aktionen der einzelnen Akteure, welche auf die Integration der Verhaltenslogik in den Digitalen Zwilling abzielen. Die Entwicklung der einzelnen Elemente des Digitalen Zwillings liegen außerhalb der Systemgrenze und werden nicht betrachtet. Es wird angenommen, dass die im Rahmen der Produktentwicklung entstehenden Modelle und Software wiederverwendet und im Rahmen des Anwendungsfalls importiert und miteinander verknüpft werden.

Systemakteure

Die beteiligten Systemakteure sind einzelne Personen oder Personengruppen. Die Aktionen der menschlichen Akteure können auch durch Systeme und Prozesse automatisiert werden. Zur Übersichtlichkeit wird im Folgenden jedoch von menschlichen Einzelpersonen ausgegangen.

Der *Softwareingenieur* hat die Aufgabe, die Schnittstellen der eingebetteten Software als Grundlage der Verhaltenslogik eines Systems, welche in den Digitalen Zwilling integriert wird, zu beschreiben. Dabei sind keine Änderungen der eingebetteten Software vorgesehen.

Diese Randbedingung resultiert daraus, dass das im Rahmen der Dissertation entwickelte Konzept auch auf bereits bestehende Systeme angewendet werden soll und dabei die identische eingebettete Software verwendet wird.

Der *Elektronikingenieur* ist insbesondere für die Entwicklung der eingebetteten Hardware zuständig. Er bildet die Schnittstelle zwischen dem *Softwareingenieur* und dem *System- und Konstruktionsingenieur*, da die eingebettete Hardware die Softwaredomäne mit der Mechanik verbindet. Der *Elektronikingenieur* bezieht die notwendigen Modelle und das Fachwissen aus der Entwicklung der eingebetteten Hardware und setzt geeignete Softwarewerkzeuge und Integrierte Entwicklungsumgebungen der Elektronikentwicklung ein, um die eingebettete Software virtuell ausführen zu können.

Der *Systemingenieur* übernimmt die zentrale Aufgabe der Gestaltung der Schnittstelle zwischen der Software und Hardware des Systems mit den mechanischen Komponenten. Dazu definiert er einen Anknüpfungspunkt zwischen den unterschiedlichen Produktentwicklungsdomänen. Der *Systemingenieur* übernimmt im Rahmen des vorgestellten Anwendungsfalls auch die Aufgabe der Instanziierung des Digitalen Zwillings. Diese Aktivität gehört in der aktuellen Praxis nicht zu den typischen Aufgaben eines *Systemingenieurs*. Aufgrund der interdisziplinären Aufgabe werden disziplinübergreifende Fähigkeiten benötigt.

Die *Anforderungen Datenbank* beinhaltet die Anforderungen des Kunden, für den der Digitale Zwilling mit integrierter Verhaltenslogik erstellt wird. Der Kunde bestimmt die Ausprägung und den Fokus des Digitalen Zwillings. Wie im Stand der Technik dargelegt, ist ein Digitaler Zwilling stets auf einzelne Aufgaben ausgelegt und kann nicht Anforderungen aus allen Anwendungsfällen gleichzeitig erfüllen. Somit werden Vorgaben benötigt, die durch den Kunden als späterer Nutzer des Digitalen Zwillings definiert und in einer Datenbank abgelegt wurden, um die Zusammensetzung des Digitalen Zwillings auszulegen.

Der *Konstrukteur* übernimmt die Bereitstellung und Integration der mechanischen Modelle des Produkts in den Digitalen Zwilling. Dazu werden Modelle aus der Produktentwicklung verwendet, um daraus die für den Anwendungsfall notwendigen Elemente des Digitalen Zwillings zu generieren. Dabei stützt sich der Konstrukteur auf die Anforderungen des Kunden und orientiert sich an den Definitionen der Architektur und Schnittstellen. Abhängig vom Anwendungsfall werden die Fähigkeiten eines *Simulationsingenieurs* benötigt, weshalb dieser statt des *Konstrukteurs* bzw. diesen ergänzend auftreten kann.

Aktionen und Beziehungen

Die erläuterten Akteure sind den Aktivitäten des Anwendungsfalls zugeordnet. Jede Aktivität kann in einem eigenständigen Anwendungsfall konkreter spezifiziert werden. Die Aktivitäten weisen gegenseitige Abhängigkeiten auf, welche durch Beziehungen aufgezeigt werden. Im Folgenden werden die Aktivitäten und die Beziehungen eingehend beschrieben.

Anforderungen berücksichtigen: Die Grundlage der Ausprägung des Digitalen Zwillings bilden die Anforderungen des Kunden, die in einer *Anforderungen Datenbank* abgelegt sind und im beschriebenen Anwendungsfall mittels Abhängigkeiten berücksichtigt werden.

Schnittstelle der eingebetteten Software beschreiben: Die Verhaltenslogik eines eingebetteten Systems wird durch die eingebettete Software bestimmt. Diese steuert das System, leitet den Zustand der Sensoren des Systems über Schnittstellen an andere Systeme und den Menschen weiter und reagiert auf externe Eingaben. Die eingebettete Software wird speziell für ein konkretes eingebettetes System entwickelt bzw. konfiguriert und interagiert mit der Mechanik des Systems mittels der eingebetteten Hardware, auf welcher die Software ausgeführt wird. Während der Entwicklung der eingebetteten Software werden die Schnittstellen zwischen Software und Hardware entsprechend der Architektur des Mikrocontrollers und dem Aufbau der Elektronik festgelegt. Damit wird bestimmt, wie die Software mit der Hardware interagiert. Diese Schnittstellendefinition der Software ist notwendig, um eine geeignete Schnittstelle auszulegen. Während der Aktivität entnimmt der Softwareingenieur eine Schnittstellenbeschreibung aus der Dokumentation der Software oder analysiert die bestehende Softwareimplementierung und entwickelt daraus die Schnittstellendefinition.

Elektroniksimulation integrieren: In dieser Aktion bindet der *Elektronikingenieur* die bereits vorliegenden Elektroniksimulation an die definierte Architektur und die Schnittstellen an. Damit steht die virtuelle Elektronik im Digitalen Zwilling zur Verfügung, um die eingebettete Software zu integrieren. Der Detailgrad einer Elektroniksimulation muss vom *Elektronikingenieur* in Abhängigkeit der Kundenanforderungen gewählt werden. In Analogie zu anderen Simulationsarten muss dabei zwischen Ausführungsgeschwindigkeit und Genauigkeit der Simulation abgewogen werden.

Architektur und Schnittstellen festlegen: Der *Systemingenieur* hat die Aufgabe, die notwendige Architektur und Schnittstellen im Digitalen Zwilling festzulegen. Mit dem Fokus auf die Integration des Verhaltens in den Digitalen Zwilling orientiert er sich dabei an den Vorgaben durch den *Softwareingenieur* und die Anforderungen des Kunden. Damit übernimmt der *Systemingenieur* eine zentrale Aktion bei der Integration der Verhaltenslogik in den Digitalen Zwilling, da dieser auch bei der Produktentwicklung eine Schnittstelle zwischen den einzelnen Domänen bildet.

Modelle aus Entwicklung in Digitalen Zwilling überführen: Bei der Entwicklung des Digitalen Zwillings können Modelle aus der Produktentwicklung verwendet werden. Diese Modelle müssen durch einen *Konstrukteur* in die Domäne des Digitalen Zwillings überführt werden, um diese im Rahmen des beschriebenen Anwendungsfalls nutzbar zu machen. Die Art und Detailgrad der Modelle sind dabei anhängig von den Anforderungen, welche durch die *Anforderungen Datenbank* bereitgestellt werden.

Mechanische Simulationen integrieren: Die aus der Produktentwicklung überführen Simulationen müssen anschließend mittels definierter Schnittstellen an den Digitalen Zwilling angebunden werden. Diese Aktion wird ebenfalls vom *Konstrukteur* durchgeführt und ist abhängig von den Modellen aus der Produktentwicklung.

Digitalen Zwilling mit integrierter Verhaltenslogik instanziieren: Die zentrale Aktion des beschriebenen Anwendungsfalls, welche direkt oder indirekt von den Aktionen abhängt, kann als Auslöser des Anwendungsfalls betrachtet werden. Alle Aktivitäten resultieren daraus, dass der *Systemingenieur* einen Digitalen Zwilling mit integrierter Verhaltenslogik instanziieren kann und damit die Grundlage der späteren Nutzung legt.

Das Ergebnis des Anwendungsfalls ist ein Digitaler Zwilling mit integrierter Verhaltenslogik, welcher eine geeignete Architektur und Schnittstellen nutzt, um die einzelnen Elemente aus unterschiedlichen Domänen miteinander zu verknüpfen. Basierend auf dieser Vorarbeit können anschließend diverse Anwendungsfälle umgesetzt werden, welche die Abbildung des Verhaltens im Digitalen Zwilling voraussetzen.

3.3.2 Steuerung des Digitalen Zwillings

Im nachfolgenden Abschnitt wird der Anwendungsfall der Steuerung des Digitalen Zwillings beschrieben. Dieser Anwendungsfall wird erst möglich, indem vorher die verhaltensabbildende Logik bereits integriert wurde. Nachdem das geschehen ist, kann ein *Nutzer* den Digitalen Zwilling in analoger Weise zum Physischen Zwilling steuern. Die Abbildung 3.2 visualisiert diesen Anwendungsfall. Das Anwendungsfalldiagramm berücksichtigt neben dem Nutzer die drei Akteure *Externe Datenbank*, *Simulationsserver* und *Physischer Zwilling*. Im Folgenden werden die einzelnen Aktionen, die Akteure sowie die Systemgrenze näher erläutert.

Systemgrenze

Die Systemgrenze umfasst alle Aktivitäten, die im Rahmen des Digitalen Zwillings durchgeführt werden müssen, um die Steuerung des Digitalen Zwillings zu ermöglichen. Dabei liegen einige Funktionen des Digitalen Zwillings, wie beispielsweise die Bereitstellung von

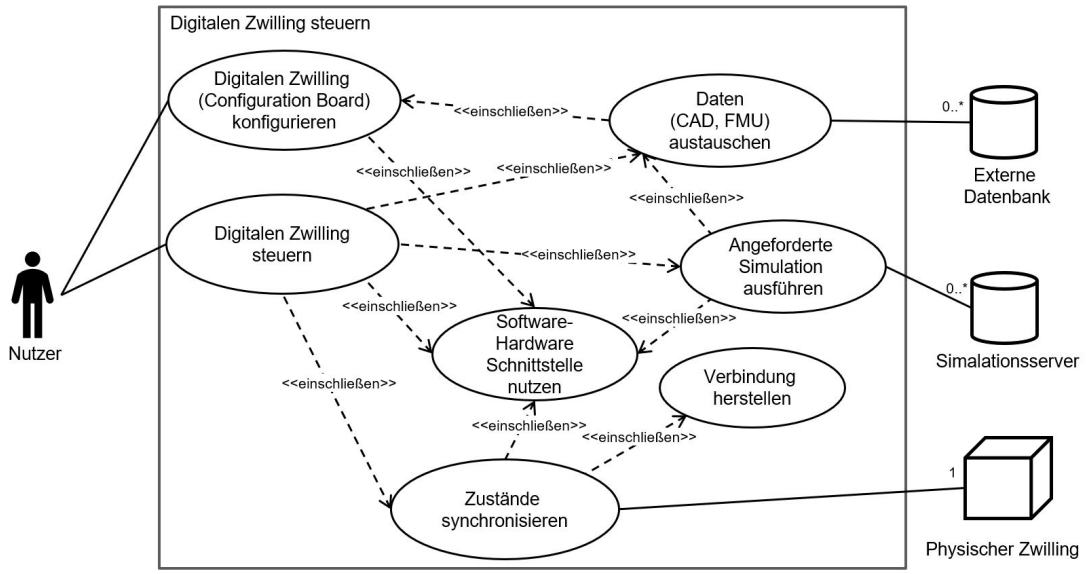

Abbildung 3.2: Anwendungsfalldiagramm zur Steuerung des Digitalen Zwillings

Daten durch eine *Externen Datenbank* sowie die Durchführung von Simulationen mittels externen *Simulationsserver*, außerhalb der betrachteten Systemgrenze.

Systemakteure

Die am Anwendungsfall beteiligten Systemakteure sind Personen und externe Systeme. Das in der Abbildung 3.2 dargestellte Anwendungsfalldiagramm weist die vier Systemakteure *Nutzer*, *Externe Datenbank*, *Simulationsserver* und *Physischer Zwilling* auf.

Der *Nutzer* ist im betrachteten Anwendungsfall eine menschliche Person, die den Digitalen Zwilling auf die gleiche Art und Weise steuern möchte, wie sie auch den Physischen Zwilling steuern kann. Die Motivation kann beispielsweise darin liegen, dass mittels des Digitalen Zwillings einzelne Eigenschaften des Verhaltens des Physischen Zwillings beobachtet und analysiert werden sollen. Bevor der *Nutzer* jedoch den Digitalen Zwilling steuern kann, muss die gewünschte Zusammensetzung der einzelnen Elemente des Digitalen Zwillings definiert werden, indem eine Konfiguration dessen durchgeführt wird.

Die *Externe Datenbank* ist für die flexible Bereitstellung von der zum Betrieb des Digitalen Zwillings erforderlichen Daten zuständig. Unter der Voraussetzung eines verteilten Digitalen Zwillings können es je nach Konfiguration keine oder mehrere *Externe Daten-*

banken sein, auf denen die Daten vorliegen. Vereinfachend wurden diese Datenbanken unter einem Akteur zusammengefasst und durch die Multiplizität *0..** gekennzeichnet.

Ein *Simulationsserver* führt angeforderte Simulationen des Digitalen Zwillings aus. Bei einem verteilten Digitalen Zwilling kann sich die Ausführung je nach Konfiguration auch auf mehrere *Simulationsserver* verteilen. Dies wurde im Anwendungsfalldiagramm mit der Multiplizität *0..** berücksichtigt.

Der *Physische Zwilling* und der Digitale Zwilling bilden durch die bidirektionale Verbindung ein Zwillingssystem. Zu diesem Zweck muss zwischen den beiden Zwillingen eine Verbindung hergestellt werden. Im Betrieb können dadurch die Zustände synchronisiert werden. Je nach Intention des *Nutzers* kann im Rahmen des vorgestellten Anwendungsfalls die Synchronisation erlaubt oder blockiert werden.

Aktionen und Beziehungen

Nach der vorangegangenen Erläuterung der Akteure werden nun die einzelnen Aktionen und deren Beziehungen untereinander näher beschrieben.

Digitalen Zwilling steuern: Der Fokus des in der Abbildung 3.2 dargestellten Anwendungsfalldiagramms ist die Steuerung des Digitalen Zwillings durch den *Nutzer*. Diese Hauptaktion besitzt jedoch Abhängigkeiten zu nachgelagerten Aktionen, welche durch *«einschließen»* Beziehungen im Diagramm visualisiert sind. Die Hauptaktion wird durch den *Nutzer* ausgelöst und kann von der Datenbereitstellung und Speicherung *Externer Datenbanken* und den Diensten von *Simulationsservern* abhängen. Weiterhin wird die Aktion *Software-Hardware Schnittstelle nutzen* verwendet, um die einzelnen Elemente des Digitalen Zwillings miteinander zu verknüpfen. Falls die Synchronisation des Digitalen und Physischen Zwillings gewünscht ist, wird die *Zustände synchronisieren* Aktion ausgelöst. Die Steuerung und die Synchronisation hängen dabei von der Software-Hardware Schnittstelle ab, welche die Schnittstelle zwischen den einzelnen Elementen des Digitalen Zwillings und auch zum Physischen Zwilling darstellt.

Digitalen Zwilling konfigurieren: Die Steuerung des Digitalen Zwillings setzt voraus, dass dieser vorher konfiguriert wird. Diese Aktion wird durch den *Nutzer* durchgeführt und weist Abhängigkeiten von den Aktionen *Software-Hardware Schnittstelle nutzen* und *Daten austauschen* auf. Die Konfiguration ist notwendig, da wie im Stand der Technik dargelegt, ein Digitaler Zwilling nicht dazu geeignet ist, alle Eigenschaften eines komplexen Systems abzubilden, sondern sich nur auf begrenzte Funktionen fokussieren kann, die in einem Kontext für den Nutzer von Interesse sind.

Software-Hardware Schnittstelle nutzen: Die Konfiguration des Digitalen Zwillings schließt die Nutzung einer virtuellen Software-Hardware Schnittstelle mit ein, da diese als Schnittstelle zwischen den Elementen des Digitalen Zwillings wirkt. Veränderungen in der

Struktur des Digitalen Zwillings haben somit unmittelbaren Einfluss auf diese Schnittstelle, welche gemäß den Randbedingungen unterschiedlich verwendet wird. Diese weist als zentrale Schnittstelle weiterhin Abhängigkeiten von den Aktionen *Digitalen Zwilling steuern*, *Angeforderte Simulationen ausführen* und *Zustände synchronisieren* auf.

Daten austauschen: Der Betrieb und die Steuerung des Digitalen Zwillings hängt von Daten und Informationen sowohl des *Physischen Zwillings* als auch des Digitalen Zwillings ab, die bereitgestellt und gespeichert werden müssen. Insbesondere konfigurationsabhängige Daten müssen von *Externen Datenbanken* bezogen werden. Somit hängt diese Aktion von *Digitalen Zwilling konfigurieren* ab. Weiterhin wird die Bereitstellung und Speicherung der Daten von *Digitalen Zwilling steuern* und *Angeforderte Simulation ausführen* eingeschlossen, da diese Aktionen von externen Daten abhängig sind.

Angeforderte Simulation ausführen: Die Simulationen, aus den sich der Digitale Zwilling zusammensetzt, können von externen *Simulationsservern* ausgeführt werden. Der Akteur *Simulationsserver* bedient die Aktion *Angeforderte Simulation ausführen*, indem er Simulationsparameter und Eingabedaten aus *Daten austauschen* und *Software-Hardware Schnittstelle nutzen* bezieht und die so parametrisierte Simulation ausführt. Die Simulation findet außerhalb der betrachteten Systemgrenze statt. Die Ergebnisse der Simulation werden anschließend an die Datenbank und die virtuelle Software-Hardware Schnittstelle übergeben und weiterverarbeitet. Je nach Randbedingungen und Zusammensetzung des Digitalen Zwillings werden 0 bis n externe *Simulationsserver* eingesetzt.

Zustände synchronisieren: Zwischen dem Digitalen und Physischen Zwilling besteht eine bidirektionale Verbindung, um in beide Richtungen Daten auszutauschen und bei Bedarf die Zustände der Zwillinge zu synchronisieren. Bei den im Anwendungsfalldiagramm 3.2 beschriebenen Szenario besteht die Möglichkeit, die Steuerung des Digitalen Zwillings mit dem Physischen Zwilling zu synchronisieren oder die Synchronisation zu deaktivieren. Im adressierten Anwendungsfall wird diese Option durch die Multiplizität *0..1* der Abhängigkeit zu *Digitalen Zwilling steuern* signalisiert. Falls die Zustände synchronisiert werden sollen, wird zusätzlich die Aktion *Verbindung herstellen* ausgelöst, welche die Kommunikationsverbindung zum *Physischen Zwilling* instanziiert.

3.4 Definition des Anforderungsprofils

In diesem Abschnitt werden Anforderungen definiert, die an die einzelnen Elemente des Konzepts, der Implementierung und die Validierung gestellt werden. Dabei betreffen die Anforderungen sowohl die Konzeptionierung, als auch die Implementierung. Aus diesem Grund folgt im Rahmen der Verifikation der Anforderungen im Kapitel 6 die Beurteilung jeder einzelnen Anforderung hinsichtlich der Erfüllung im Konzept und in

der Implementierung.

Die Anforderungen resultieren einerseits aus der im Handlungsbedarf motivierten Zieldefinition mit der Einschränkung auf die betrachteten Anwendungsfälle. Andererseits muss das späteren Einsatzumfeldes berücksichtigt werden. Zu dem Einsatzumfeld gehören z.B. die Qualifikation der Zielgruppen oder beteiligte Geschäftsprozesse. Gleichzeitig muss auch das technische Umfeld betrachtet werden. Zu diesem gehören die verfügbaren und eingesetzten Soft- und Hardwaresysteme, Netzwerkverbindungen und Schnittstellen zu anderen Anwendungen. Aus diesen Bedingungen können Anforderungen abgeleitet werden. Eine Anforderung ist gemäß DIN EN ISO 9000 wie folgt definiert:

„Eine Anforderung ist ein Erfordernis oder eine Erwartung, das oder die festlegt, üblicherweise voraussetzt oder verpflichtend ist“ [43].

Diese Definition unterscheidet damit unterschiedliche Arten von Anforderungen. *Festgelegte Anforderungen* sind beispielsweise in dokumentierten Informationen enthalten. *Üblicherweise vorausgesetzte Anforderungen* werden als allgemeine Praxis betrachtet und implizit vorausgesetzt. Die Norm empfiehlt weiterhin, ein Bestimmungswort zu verwenden, um die Anforderungsarten zu spezifizieren und zu gliedern [43]. Aufgrund von begrenzten Ressourcen und möglichen Konflikten zwischen konträren Forderungen ist es in der Regel nicht möglich, alle Anforderungen in gleicher Qualität zu erfüllen [19]. Aus diesen Gründen wird im Folgenden eine Priorisierung durch Unterteilung der Anforderungen in *Forderungen* (F) und *Wünsche* (W) durchgeführt. Dabei sollen Forderungen zwingend erfüllt werden, während Wünsche nach Möglichkeit zu berücksichtigen sind [19].

Anforderungen an die eingebettete Software und deren Virtualisierung

Anforderung 1 – Integration der eingebetteten Software in den Digitalen Zwilling

Die eingebettete Software des Physischen Zwillings muss in den Digitalen Zwilling integriert werden können.

Moderne technische Systeme, insbesondere Cyber-Physischen Systeme, beinhalten Software. Die Steuerungssoftware bildet die Schnittstelle des Systems zu einem Bediener bzw. einem anderen technischen System und gibt maßgeblich das Verhalten des Systems vor. Mittels der Software werden Funktionen umgesetzt, die auch bei gleichbleibender Hardware und Mechanik einen Mehrwert bieten. In der Gesamtbetrachtung steigt der Anteil der durch Software umgesetzten Funktionen stetig an, sodass diese eine immer wichtigere Rolle einnimmt und auch bei den Herstellern der technischen Systeme in den

Fokus rückt. Aus dieser Entwicklung resultiert die Forderung, dass die Software eines Physischen Zwillings auch in den korrespondierenden Digitalen Zwilling integriert werden muss, um ein ganzheitliches virtuelles Abbild zu ermöglichen.

Anforderung 2 – Virtualisierung eingebetteter Hardware

Die eingebettete Software muss ohne physische eingebettete Hardware ausgeführt werden können.

eingebettete Software wird für eingebettete Hardware entwickelt. Die zentrale Rechnereinheit der eingebetteten Hardware bildet ein Mikroprozessor, welche für die speziellen Aufgaben des eingebetteten Systems ausgelegt ist. Die Rechnerarchitektur des Mikroprozessors entspricht in der Regel nicht der Rechnerarchitektur des Systems, welches für den Betrieb des Digitalen Zwillings zuständig ist. Deshalb ist es notwendig, mittels Hilfswerkzeuge die Virtualisierung der eingebetteten Hardware zu ermöglichen, um darin anschließend die eingebettete Software ausführen zu können.

Anforderung 3 – Gleiche Software in beiden Zwillingen

Der Digitale Zwilling soll die gleiche Software verwenden, welche auch im Physischen Zwilling eingesetzt wird.

Das Ziel eines Digitalen Zwillings ist die originalgetreue bzw. exakte Abbildung des physischen Zwillings. Dies gilt auch für die Äquivalenz im Verhalten. Auf Softwareebene kann dies gewährleistet werden, indem exakt die gleiche Software in beiden Zwillingen verwendet wird. Für den Fall, dass für den Digitalen Zwillinge die Software angepasst werden muss, müsste diese Anpassung bei jeder Veränderung in der Software wiederholt werden. Dies würde zu erhöhtem Aufwand bei Anpassungen von Veränderungen im Physischen Zwilling führen und sollte vermieden werden.

Anforderungen an die virtuelle Software-Hardware Schnittstelle

Anforderung 4 - Schnittstelle zur eingebetteten Software

Es muss eine Schnittstelle zwischen der eingebetteten Software und den davon abhängigen Modulen des Digitalen Zwillings existieren

Gemäß der Motivation der vorliegenden Dissertation und gemäß *Anforderung 1 – Integration der eingebetteten Software in den Digitalen Zwilling*, soll die eingebettete Software eines physischen Systems in dessen Digitalen Zwilling integriert werden. Als Schnittstelle zwischen der eingebetteten Software und den davon abhängigen Modulen des Digitalen Zwillings muss eine entsprechende virtuellen Software-Hardware Schnittstelle eingesetzt werden.

Anforderung 5 - Informationsmodell als Basis

Das Informationsmodell dient als Grundlage für die Generierung der Schnittstellen im Digitalen Zwilling.

Der Digitale Zwilling besitzt Schnittstellen sowohl für die bi-direktionale Verbindung zum Physischen Zwilling als auch zum Nutzer und anderen Systemen. Über die Schnittstellen müssen die Eigenschaften und Funktionen des Digitalen Zwillings, je nach Anwendung und Berechtigungsgruppe, erreichbar sein. Das Informationsmodell beschreibt eben jenen Eigenschaften und Funktionen und ist dadurch als Basis der Schnittstellen geeignet.

Anforderung 6 – Aufrüstung

Die virtuelle Software-Hardware Schnittstelle muss so konzipiert sein, dass sowohl bei einer Neuentwicklung, als auch beim Aufrüsten eines bestehenden Systems diese verwendet werden kann.

Digitale Zwillinge können sowohl bei Neuentwicklungen (Greenfield-Ansatz) bereits eingeplant werden, als auch bei bestehenden Systemen (Brownfield-Ansatz) nachgerüstet werden. Bei einem Greenfield-Ansatz wird das gesamte System auf die Zusammenwirkung innerhalb eines Zwillingssystems ausgelegt. Damit können auch die Schnittstellen bereits von Beginn der Entwicklung konzipiert werden. In einem Brownfield-Ansatz sind jedoch Randbedingungen bereits vorgegeben, die bei der Einführung von neuen Funktionen, wie beispielsweise auch Schnittstellen, berücksichtigt werden müssen. Um in beiden Ansätzen eingesetzt werden zu können, muss die Software-Hardware Schnittstelle sowohl bei Neuentwicklungen, als auch bei der Nachrüstung bestehender Systeme eingesetzt werden können.

Anforderung 7 - Erweiterbarkeit

Die virtuelle Software-Hardware Schnittstelle muss erweiterbar sein.

Eine Anforderung des Digitalen Zwillings ist die Anpassungsfähigkeit mit der sich dieser an verändernde Rahmenbedingungen und auch an Änderungen im Physischen Zwilling adaptieren kann. Die Anpassungsfähigkeit betrifft auch die interne Struktur und die Schnittstellen im Digitalen Zwilling. Daraus erwächst die Anforderung, dass die Software-Hardware Schnittstelle veränderbar, anpassbar und erweiterbar sein muss.

Anforderung 8 - Domänenübergreifender Einsatz

Die virtuelle Software-Hardware Schnittstelle muss domänenübergreifend eingesetzt werden können.

Die einzelnen Elemente des Digitalen Zwillings betreffen unterschiedliche technische Domänen und werden innerhalb dieser entwickelt. Eine der Kernaufgaben des Konzepts eines Digitalen Zwillings ist das Zusammenbringen dieser Domänen und die Ermöglichung der Wechselwirkung der einzelnen Elemente. Dabei übernimmt die Schnittstelle zwischen zwei Elemente aus unterschiedlichen Domänen die bedeutsame Aufgabe, beiden Domänen eine gemeinsame Kommunikationsgrundlage bereitzustellen. Daher muss die Software-Hardware Schnittstelle domänenübergreifend eingesetzt werden können, indem eine allgemein verständliche und nutzbare Semantik verwendet wird.

Anforderung 9 - Methodische Vorgehensweise

Bei der Erstellung einer virtuellen Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells muss eine methodische und reproduzierbare Vorgehensweise angewendet werden.

Eine Schnittstelle ist die Grundlage der Interaktion zwischen zwei Elementen und muss transparent, nachvollziehbar und reproduzierbar gestaltet sein. Dies ist die Voraussetzung, dass die Schnittstelle von den Interaktionspartnern korrekt umgesetzt und richtig eingesetzt werden kann. Daraus erwächst die Anforderung, dass die Erstellung und Veränderung einer Schnittstelle gemäß einer methodischen Vorgehensweise erfolgen muss, um diesen Vorgang nachvollziehen und korrekt reproduzieren zu können.

Anforderung 10 - Werkzeugunterstützung

Die Erstellung und Erweiterung der virtuellen Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells soll mittels geeigneter Werkzeuge unterstützt und vereinfacht werden.

Die Software-Hardware Schnittstelle und das zugrundeliegende Informationsmodell erreicht eine Komplizität, die von einem Menschen kaum bewältigt werden kann. Aus diesem Grund werden unterstützende und teilautomatisierte Werkzeuge benötigt, um die zuständige Person bei der Erstellung und Erweiterung der Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells anzuleiten und zu entlasten.

Anforderungen an den Digitalen Zwilling

Im Rahmen des zu entwickelnden Konzepts ist es von Bedeutung, dass ein prototypischer Digitaler Zwilling eingesetzt wird, welcher die wichtigsten, in Kapitel 2.3.2 beschriebenen, Charakteristika erfüllt. Resultierend daraus werden im nachfolgenden Abschnitt einige bedeutsame Anforderungen an den Digitalen Zwilling definiert.

Anforderung 11 – Bidirektionale Verbindung

Der Physische und der Digitale Zwilling müssen eine bidirektionale Verbindung besitzen.

Der Digitale Zwilling muss mit dem Physischen Zwilling über eine Kommunikationsschnittstelle verbunden werden können mittels derer eine bidirektionale Kommunikation und Interaktion ermöglicht wird.

Anforderung 12 – Werkzeugneutralität

Der Digitale Zwilling soll ohne domänenspezifische Autorenwerkzeuge einsetzbar sein.

Das Konzept eines Digitalen Zwillings zielt auf den begleitenden Einsatz in Nutzungsphase eines physischen Produkts ab. Der Digitale Zwilling soll den Physischen Zwilling im Betrieb virtuell abbilden. Der Betreiber bzw. der Nutzer des Physischen Zwillings ist in diesen Anwendungsfällen somit auch der Nutzer des Digitalen Zwillings. Aus diesen Rahmenbedingungen wird abgeleitet, dass der Digitale Zwilling möglichst ohne domänenspezifischer Autorenwerkzeuge betrieben werden muss, da der Nutzer weder die Werkzeuge noch das domänenspezifische Wissen für den Einsatz dieser Werkzeuge besitzt. Stattdessen sollen neutrale und weit verfügbare Softwarewerkzeuge eingesetzt werden, um die Zugänglichkeit zu dem Digitalen Zwilling zu vergrößern.

Anforderung 13 - Weiterentwicklung in der Nutzungsphase

Der Digitale Zwilling muss auch in der Nutzungsphase weiterentwickelt werden können.

Der Einsatz des Digitalen Zwillings erstreckt sich über mehrere Produktlebensphasen. Ein Produkt, wie z.B. ein Cyber-Physisches System, kann sich während der Nutzung wesentlich verändern und inkrementell weiterentwickeln. Dies macht es erforderlich, dass auch der Digitale Zwilling einerseits der Entwicklung des physischen Systems folgen kann. Andererseits kann auch der Funktionsumfang des Digitalen Zwillings in der Nutzungsphase immer weiter erhöht werden. Aus diesen Gründen muss der Digitale Zwilling in einem iterativen Prozess beständig weiterentwickelt werden können.

Anforderung 14 - Breite Einsetzbarkeit

Die Grundstruktur des Digitalen Zwillings sollte auf unterschiedliche Plattformen übertragen werden können.

Der Digitale Zwilling wird auf digitalen Systemen ausgeführt, welche durch Einschränkungen auf technologische Randbedingungen, wie beispielsweise Betriebssysteme, Programmiersprachen und Kommunikationsprotokolle, abgegrenzte Plattformen bilden. Um eine breite Einsetzbarkeit und Unabhängigkeit zu ermöglichen, sollte die Grundstruktur des Digitalen Zwillings grundsätzlich zwischen unterschiedlichen Plattformen übertragen werden können.

Anforderung 15 - Ausführungsgeschwindigkeit

Die Ausführungsgeschwindigkeit des Digitalen Zwillings soll möglichst an das Verhalten des realen Systems angepasst sein.

Das Verhalten eines Systems wird durch Änderung des Zustandes über Zeit beschrieben. Somit muss die Zeit bei der Abbildung des Verhaltens eines Systems berücksichtigt werden. Grundsätzlich können die Elemente eines Digitalen Zwillings auf eine gemeinsame Zeitmessung synchronisiert werden, um ein korrektes Zusammenwirken zu gewährleisten. Die Simulationszeit kann dabei um ein Vielfaches höher sein, als die abgebildete (simulierte) Zeitspanne. Um jedoch eine realitätsnahe Abbildung des Physischen Systems zu gewährleisten, ist eine echtzeitnahe Ausführungsgeschwindigkeit anzustreben.

Anforderungen an die Modelle und die Konfigurationen des Digitalen Zwillings

Anforderung 16 - Modelle im neutralen Datenformat

Die im Digitalen Zwilling eingesetzten Modelle sollen in einem neutralen Datenformat vorliegen.

Die Modelle des Digitalen Zwillings können in aus Modellen der Produktentwicklung abgeleitet werden. Wie durch *Anforderung 12 - Werkzeugneutralität* gefordert, soll jedoch möglichst auf den Einsatz von Autorenwerkzeugen, in denen die Modelle erstellt wurden, verzichtet werden. Daraus erwächst die Anforderung, dass die im Digitalen Zwilling eingesetzten Modelle in einem neutralen Format vorliegen sollen.

Anforderung 17 – Konfiguration

Der Digitale Zwilling muss anwendungsbezogen konfiguriert werden können.

Wie im Stand der Technik dargestellt, kann der Digitale Zwilling in der Regel nicht alle Eigenschaften des Physischen Zwillings gleichzeitig abbilden. Dies trifft insbesondere auf komplexe Systeme zu, die unter unterschiedlichen Rahmenbedingungen agieren. Aus diesem Grund benötigt der Digitale Zwilling eine Möglichkeit zur anwendungsbezogenen Konfiguration der Zusammensetzung der einzelnen Modelle und Daten. Da diese Modelle und Daten über Schnittstellen miteinander interagieren, haben diese Schnittstellen bei der Konfiguration eine bedeutsame Funktion. Mittels der Konfiguration können individuelle und einsatzspezifische Digitale Zwillinge ausgebildet werden.

Anforderung 18 - Modularität

Der Digitale Zwilling muss modular aufgebaut sein.

Die in der *Anforderung 17 - Konfiguration* beschriebene Konfigurierbarkeit setzt voraus, dass der Digitale Zwilling modular zusammengesetzt ist. Die einzelnen, möglichst autonomen Module sollen je nach Anwendungsfall zusammengesetzt werden, um die gewünschte Funktion des Digitalen Zwillings zu erreichen.

Anforderung 19 - Reduzierung der Komplexität

Zur Reduzierung der Komplexität soll die Funktion der einzelnen Module des Digitalen

Zwillings hinter Schnittstellen verborgen werden.

Zur Reduzierung der Komplexität soll die Funktion der einzelnen Module hinter Modulschnittstelle verborgen sein, weil die Nutzung unabhängig von domänenspezifischem Wissen erfolgen soll.

Anforderung 20 - Unterstützung bei der Konfiguration

Der Anwender soll bei der Konfiguration der virtuellen Software-Hardware Schnittstelle unterstützt werden.

Die Konfiguration erfolgt in der Nutzungsphase des Systems und ermöglicht die flexible Zusammensetzung des Digitalen Zwillings. Falsche Konfiguration kann dabei zu Ausfällen des Digitalen Zwillings und auch des Physischen Systems führen. Um die Fehleranfälligkeit zu verbessern, soll der Anwender während der Konfiguration unterstützt werden, indem inkorrekte Konfigurationen automatisch detektiert und verhindert werden.

Tabelle 3.1: Anforderungsprofil

Nr.	Art	Bezeichnung
Anforderungen an die eingebettete Software		
1	F	*Die eingebettete Software des Physischen Zwillings muss in den Digitalen Zwilling integriert werden können.*
2	F	*Die eingebettete Software muss ohne physische eingebettete Hardware ausgeführt werden können.*
3	W	*Der Digitale Zwilling soll die gleiche Software verwenden, welche auch im Physischen Zwilling eingesetzt wird.*
Anforderungen an die virtuelle Software-Hardware Schnittstelle		
4	F	*Es muss eine Schnittstelle zwischen der eingebetteten Software und den davon abhängigen Modulen des Digitalen Zwillings existieren.*
5	F	*Das Informationsmodell dient als Grundlage für die Generierung der Schnittstelle im Digitalen Zwilling.*
6	F	*Die virtuelle Software-Hardware Schnittstelle muss so konzipiert sein, dass sowohl bei einer Neuentwicklung, als auch beim Aufrüsten eines bestehenden Systems diese verwendet werden kann.*
7	F	*Die virtuelle Software-Hardware Schnittstelle muss erweiterbar sein.*
8	F	*Die virtuelle Software-Hardware Schnittstelle muss domänenübergreifend eingesetzt werden können.*
9	F	*Bei der Erstellung einer virtuellen Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells muss eine methodische und reproduzierbare Vorgehensweise angewendet werden.*
10	F	*Die Erstellung und Erweiterung der virtuellen Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells soll mittels geeigneter Werkzeuge unterstützt und vereinfacht werden.*

Tabelle 3.2: Anforderungsprofil (Fortsetzung)

Nr.	Art	Bezeichnung
Anforderungen an den Digitalen Zwilling		
11	F	*Der Physische und der Digitale Zwilling müssen eine bidirektionale Verbindung besitzen.*
12	W	*Der Digitale Zwilling soll ohne domänenspezifische Autorenwerkzeuge einsetzbar sein.*
13	F	*Der Digitale Zwilling muss auch in der Nutzungsphase weiterentwickelt werden können.*
14	W	*Die Grundstruktur des Digitalen Zwillings sollte auf unterschiedliche Plattformen übertragen werden können.*
15	W	*Die Ausführungsgeschwindigkeit des Digitalen Zwillings soll möglichst an das Verhalten des realen Systems angepasst sein.*
Anforderungen an die Modelle und die Konfiguration		
16	W	*Die im Digitalen Zwilling eingesetzten Modelle sollen in einem neutralen Datenformat vorliegen.*
17	F	*Der Digitale Zwilling muss anwendungsbezogen konfiguriert werden können.*
18	F	*Der Digitale Zwilling muss modular aufgebaut sein.*
19	W	*Zur Reduzierung der Komplexität soll die Funktion der einzelnen Module des Digitalen Zwillings hinter Schnittstellen verborgen werden.*
20	W	*Der Anwender soll bei der Konfiguration der virtuellen Software-Hardware Schnittstelle unterstützt werden.*

4 Konzept der Virtuellen Klemmleiste

Im nachfolgenden Kapitel wird basierend auf den hergeleiteten Anforderungen das Konzept einer Virtuellen Klemmleiste als eine virtuelle Software-Hardware Schnittstelle zur Integration der Verhaltenslogik in den Digitalen Zwilling entwickelt. Dazu wird zunächst eine Konzeptübersicht vorgestellt, welche auch die konzeptionelle Herangehensweise beinhaltet.

4.1 Konzeptübersicht

Das Fundament des Konzepts bildet der im Kapitel 2 beschriebene Stand der Technik auf den Gebieten der Digitalisierung, der Industrie 4.0, des Digitalen Zwillings und der Simulation. Aus den Anwendungsfällen der Industrie 4.0 und des Digitalen Zwillings leitet sich ein Handlungsbedarf ab, welcher im Kapitel 3 definiert wurde. Der Stand der Technik sowie die ausgewählten Anwendungsfälle bestimmen die Anforderungen an das Konzept und wurden im Kapitel 3.4 zusammengetragen. Auf dieser Basis wird in folgenden Kapiteln ein Konzept erarbeitet, welches die Anwendungsfälle ermöglicht und dabei die Anforderungen erfüllt.

Abbildung 4.1: Konzeptionellen Schritte bei der Integration der Verhaltenslogik in den Digitalen Zwilling mittels einer Virtuellen Klemmleiste

Die Abbildung 4.1 zeigt die konzeptionellen Schritte bei der Integration der Verhaltenslogik in den Digitalen Zwilling. Zunächst ist es notwendig, die Verhaltenslogik genauer zu definieren. Anschließend wird ein Vorgehen entwickelt, wie diese Verhaltenslogik im Digitalen Zwilling repräsentiert werden kann, indem diese mittels geeigneter Werkzeuge

virtualisiert wird. Zur Anbindung der virtuellen Verhaltenslogik an die anschließenden Modelle des Digitalen Zwillings wird im nächsten Schritt die Virtuelle Klemmleiste generiert. Die drei ersten Konzeptbausteine ermöglichen den letzten Prozessschritt, in dem die Erweiterung des Digitalen Zwillings durch die Verhaltenslogik durchgeführt wird. Die für den Gesamtprozess erforderlichen Vorgehensweisen werden als eine Prozessbeschreibung erarbeitet, die als Leitfaden herangezogen werden kann. Das Vorgehen ist in der Abbildung 4.2 als Gesamtübersicht dargestellt und wird im Folgenden zunächst auf einer übergeordneten Abstraktionsebene erläutert, bevor weiter in die einzelnen Teilabschnitte untergliedert wird.

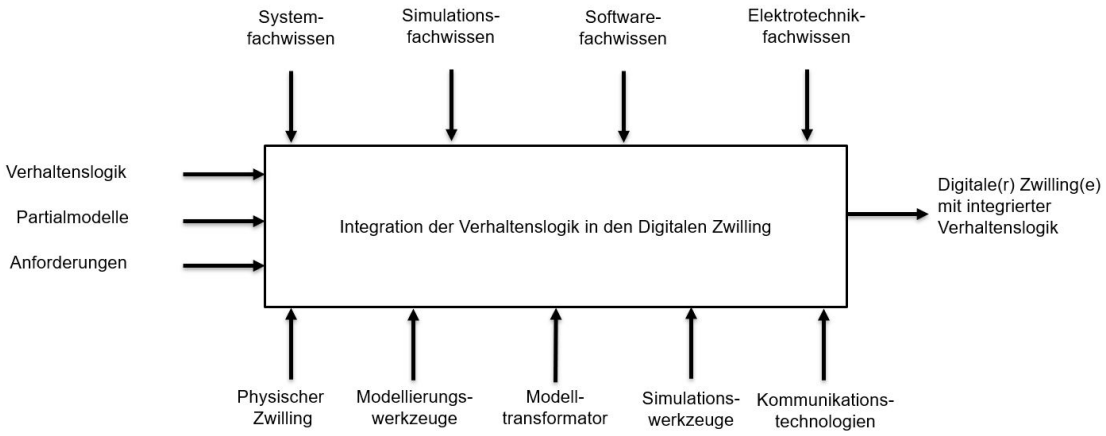

Abbildung 4.2: Konzeptübersicht

Für eine strukturierte und standardisierte Darstellung der Vorgehensweise werden die Prozesse mittels der *Structured Analysis and Design Technique* (SADT) - Notation modelliert. Diese Notation bietet im vorliegenden Fall mehrere Vorteile. Die hierarchische Struktur erlaubt die einfache Unterteilung des Gesamtprozesses ist Teilprozesse, die auf eigenen Ebenen weiter ausgeführt werden. Zudem ist die Notation einfach zu erlernen und zu erklären, weshalb diese für den interdisziplinären Prozess mit Beteiligten aus unterschiedlichen Domänen gut geeignet ist. SADT ist eine graphische Modellierungsmethodik für Aktivitäten und Datenflüsse, die Mittels hierarchischer Ebenen in einem Top-Down-Verfahren einen Prozess schrittweise und in verschiedenen Darstellungen konkretisiert. Die A-0 Ebene stellt dabei die oberste Ebene dar. Die Eingabedaten einer Aktivität werden durch beschriftete Pfeile von der linken Seite kommend dargestellt. Die Ausgaben einer Aktivität werden auf der rechten Seite platziert. Die zur Durchführung der Aktivität notwendigen Mechanismen und Ressourcen fließen von unten ein. Die Steuerungsdaten werden

von oben kommend aufgetragen. Im Kontext der Dissertation wird auch das Fachwissen aus unterschiedlichen Disziplinen als eine Steuerungsfunktion betrachtet, da dieses den Prozess mitbestimmt.

Gemäß der beschriebenen SADT-Notation wird im Folgenden die Konzeptübersicht auf der A-0 Ebene erläutert. Die Gesamtaktivität der *Integration der Verhaltenslogik in den Digitalen Zwilling* benötigt als Eingaben die *Verhaltenslogik*, die *Partialmodelle* des Digitalen Zwillings und die *Anforderungen*. Die Verhaltenslogik beschreibt dabei das Verhalten eines technischen Systems. Darunter werden insbesondere die Steuerungssoftware und die Elektronik verstanden, die das aktive Verhalten des Systems bestimmt. Unter Partialmodellen werden System- und Simulationsmodelle verstanden, die zur Abbildung der einzelnen Komponenten und Eigenschaften des Systems in einem konkreten Anwendungsfall notwendig sind. Dies können beispielsweise Simulationsmodelle der Mechanik, der Elektronik und Elektrik, der Thermodynamik usw. sein. Die tatsächliche Konstellation der Partialmodelle hängt dabei von den ebenfalls als Eingabe einfließenden Anforderungen ab, da der Digitalen Zwilling für definierte Szenarien entwickelt und ausgelegt wird.

Zur Durchführung der Aktivität sind die Ressourcen *Physischer Zwilling*, *Modellierungs-* und *Simulationswerkzeuge* sowie *Kommunikationstechnologien* notwendig. Der Physische Zwilling wird dabei benötigt, da der Digitale Zwilling per Definition stets in einer bidirektionalen Verbindung mit diesem stehen muss. Aus diesem Grund ist es notwendig, den Physischen Zwilling bei der Integration der Verhaltenslogik zu berücksichtigen. So können beispielsweise in prädiktiven Anwendungsfällen die Ausgaben der Verhaltenslogik des Digitalen Zwillings verwendet werden, um auf das Verhalten des Physischen Zwillings unter simulierten Rahmenbedingungen zu schließen. Umgekehrt sind für die regelungstechnischen Funktionen der Verhaltenslogik die aktuellen Zustandsdaten des Physischen Zwillings erforderlich. Zur virtuellen Abbildung des Systems werden *Modellierungs-* und *Simulationswerkzeuge* herangezogen. Diese können getrennt oder in einer gemeinsamen Umgebung vorliegen. Durch die Eigenschaft eines Digitalen Zwilling, permanent im Betrieb zu sein und eine möglichst echtzeitnahe Ausführung zu erreichen, werden langfristig die Simulationswerkzeuge optimiert und fokussiert auf die Aufgabe ausgelegt werden müssen. Aus diesem Grund wurde bei der Konzeptentwicklung die Modellierungsfunktion bereits als ein eigenständiges Werkzeug betrachtet. Mittels der Modellierungswerkzeuge werden nicht nur die Simulationsmodelle erstellt, sondern auch etwaige Informationsmodelle, die sowohl für die interne als auch externe Kommunikation notwendig sind, modelliert. Als weitere Ressourcen werden außerdem *Kommunikationstechnologien* benötigt, welche die Interaktion zwischen einzelnen Teilen des Digitalen Zwillings und auch für die bidirektionale Kommunikation zum Physischen Zwilling ermöglichen.

Die Integration der Verhaltenslogik in den Digitalen Zwilling ist ein interdisziplinärer Prozess, welches Fachwissen aus den unterschiedlichen Disziplinen erfordert, um den

Integrationsprozess durchzuführen. Als Quelle des Fachwissens wird im Rahmen der vorliegenden Dissertation ein menschlicher Experte betrachtet, wobei eine Einzelperson auch mehrere Disziplinen beherrschen kann. Möglichkeiten zur Automatisierung der einzelnen Tätigkeiten des jeweiligen Experten durch Algorithmen sind möglich, werden im Rahmen des Konzepts jedoch nicht ausführlich betrachtet. Zur Durchführung wird *System-*, *Simulations-*, *Software-* und *Elektrotechnikfachwissen* benötigt, welches von einem zugehörigen Experten beigetragen werden muss. Der Systemexperte kennt das physische System und versteht die Zusammenhänge zwischen den einzelnen Funktionen, Baugruppen und Bauteilen der betrachteten Entität. Er kennt die Wechselwirkungen in Form von Energie-, Stoff- und Datenflüssen sowohl innerhalb des Systems als auch zur Außenwelt und kann diese in einem geeigneten Modell festhalten bzw. einem Dritten beschreiben, welcher die Informationen in ein Modell umsetzen kann. Der Systemexperte nimmt somit eine bedeutsame Funktion in der Koordination der einzelnen Teilmodelle des Digitalen Zwillings ein. Alternativ kann das Systemfachwissen aus der zugehörigen Dokumentation entnommen werden, wobei jedoch nicht vorausgesetzt werden kann, dass diese zum aktuellen Zeitpunkt alle notwendigen Informationen vollständig enthält, da technische Dokumentation bisher nicht darauf ausgelegt ist, ein virtuelles Modell zu generieren. Weiter wird Simulationsfachwissen eingesetzt, um die einzelnen Partialmodelle des Digitalen Zwillings zu generieren. In der Rolle des Simulationsexperten können sich sowohl Konstrukteure als auch weitere Fachleute aus der CAx - Prozesskette wiederfinden, da die Partialmodelle in Abhängigkeit vom betrachteten System, den Anforderungen und den Anwendungsfällen betrachtet werden müssen. Obwohl die Software ebenfalls als ein Partialmodell des Digitalen Zwillings betrachtet werden kann, wird es im Konzept gesondert hervorgehoben, da der Fokus auf der Verhaltenslogik liegt. Das Softwarefachwissen wird benötigt, um die Funktionen und insbesondere die Eingaben und Ausgaben der Software zu identifizieren und in die Definition einer Schnittstelle einfließen zu lassen. Das Softwarefachwissen kann ebenfalls aus einer vollständigen Dokumentation entnommen werden, falls diese vorliegt und auch verstanden und interpretiert werden kann. Als vierte Kompetenz wird Elektronikfachwissen vorausgesetzt, um das Konzept durchführen zu können. Der Elektronikexperte hat die Aufgabe, die Verbindung zwischen der Verhaltenslogik und den physischen Elementen des Systems virtuell nachzubilden. Die virtuelle Elektronik stellt anschließend die Grundlage zur Integration der eingebetteten Software in den Digitalen Zwilling.

Die Ausgabe der Aktivität und somit das Ergebnis des Gesamtprozesses ist eine oder mehrere Instanzen eines Digitalen Zwillings mit integrierter Verhaltenslogik.

Die in Abbildung 4.2 beschriebene Konzeptstruktur wird in einer detaillierteren Betrachtung in zwei Teilprozesse unterteilt, siehe Abbildung 4.3.

Diese Betrachtung unterteilt den übergeordneten Prozess in die beiden Aktivitäten

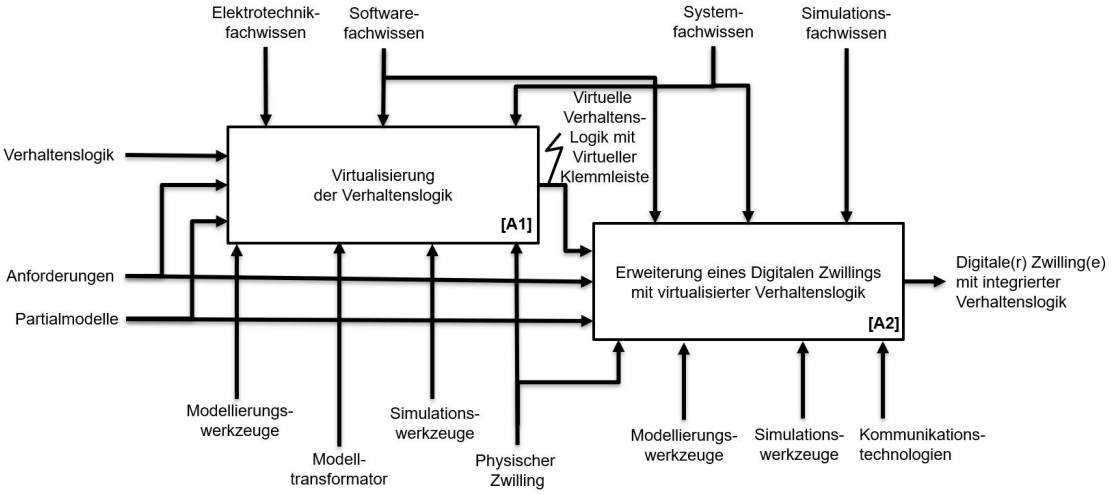

Abbildung 4.3: Aufteilung des Konzepts in zwei Teilprozesse

der *A1: Virtualisierung der Verhaltenslogik* und die *A2: Erweiterung des Digitalen Zwillings durch virtualisierte Verhaltenslogik*. In der Virtualisierung der Verhaltenslogik wird aus der Verhaltenslogik des physischen Systems eine virtuelle Verhaltenslogik generiert, die im nachfolgenden Prozess eingesetzt wird, um einen Digitalen Zwilling zu erweitern. Im Teilprozess *A1* werden neben den Anforderungen auch das Fachwissen der Software, der verwendeten Elektronik und des Gesamtsystems benötigt. Diese werden eingesetzt, um mittels Modellierungswerkzeuge eine Simulationsumgebung der Elektronik zu schaffen, die anschließend in einer Simulation als Grundlage für die Ausführung der Verhaltenslogik und die Virtuelle Klemmleiste bereitsteht. In dieser Aktivität wird der Physische Zwilling bereits berücksichtigt, da er die Virtuelle Klemmleiste, die nach der Virtualisierung der Verhaltenslogik generiert wird, maßgeblich mitbestimmt.

Die virtuelle Verhaltenslogik fließt als Ausgabe des Teilprozesses A1 in den zweiten Teilprozess A2 ein. In diesem Teilprozess wird die virtuelle Verhaltenslogik in den Digitalen Zwilling integriert, bzw. ein bestehender Digitaler Zwilling damit erweitert. In diesen Teilprozess müssen weitere Anforderungen an den Digitalen Zwilling berücksichtigt werden, die aus den angestrebten Anwendungsfällen, dem Physischen Zwilling und der allgemeinen Definition des Digitalen Zwillings resultieren. Weiterhin werden im A2 Partialmodelle des Digitalen Zwillings als Eingabe benötigt. Der Fokus dieses Teilprozesses liegt stattdessen auf einer sachgerechten Verknüpfung der Teilmodelle mittels der Informationsmodelle und Kommunikationsarchitekturen und -technologien, welche die

besonderen Anforderungen der virtuellen Verhaltenslogik erfüllen müssen. Zur Verknüpfung der einzelnen Teilmodelle werden dabei das Fachwissen über das System sowie zu den einzelnen Simulationselementen benötigt. Die jeweiligen Experten nutzen die Modellierungs- und Simulationswerkzeuge, um die Teilmodelle und das gesamte System in Anlehnung an den Physischen Zwilling zu modellieren und zu verknüpfen. Für die Interaktion der einzelnen Teilmodelle werden außerdem Kommunikationstechnologien als Ressource herangezogen. Die Ausgabe des zweiten Teilprozesses uns somit auch des Gesamtprozesses ist eine oder mehrere Instanzen eines Digitalen Zwillings mit integrierter Verhaltenslogik.

Die beiden vorgestellten Teilprozesse werden in den Kapiteln 4.3 und 4.4 eingehend erläutert. Zusammenfassend wurde in diesem Kapitel eine Übersicht der Integration der Verhaltenslogik in den Digitalen Zwilling mittels einer Virtuellen Klemmleiste geboten, die den Kern der Dissertation umreißt.

4.2 Definition der Verhaltenslogik eines Digitalen Zwillings

Bevor der Prozess der Integration der Verhaltenslogik in den Digitalen Zwilling näher beschrieben wird, erfolgt eine Definition und Eingrenzung, was unter der Verhaltenslogik von technischen Systemen im Rahmen der Dissertation verstanden wird.

Das Verhalten eines technischen Systems bildet insbesondere die Veränderungen des Systemzustandes über die Zeit ab. Die Logik beschreibt dabei die Prinzipien, nach den die Verhaltensänderung durch das System ausgeführt wird. Zum besseren Verständnis der Verhaltenslogik eines Digitalen Zwillings soll das Verständnis der Begriffe *Verhalten* und *Logik* im Kontext der vorliegenden Dissertation hergeleitet werden. Im Folgenden wird zunächst betrachtet, wie Verhalten und Logik im Allgemeinen definiert sind und wie die Begriffe auf technische Systeme übertragen bzw. angewendet werden können. Das Deutsche Wörterbuch *Duden* liefert folgende kurze Definition von Verhalten:

„*Art und Weise, wie sich ein Lebewesen, etwas verhält*“ [24].

Minton und Kahle [101] leiten im Englischen jedoch eine präzisere Definition von *Behaviour* (*dt.:* Verhalten) her:

„*Unter Verhalten versteht man die Handlungen von Individuen, Organismen, Systemen oder künstlichen Entitäten in Verbindung mit sich selbst oder ihrer Umwelt, zu der auch andere Systeme oder Organismen in der Umgebung sowie die (unbelebte) physische Umwelt gehören. Es ist die berechnete Reaktion des Systems oder Organismus auf verschiedene Reize*

oder Eingaben, ob intern oder extern, bewusst oder unbewusst, offen oder verdeckt, freiwillig oder unfreiwillig“ übersetzt aus [101].

Diese Definition von Verhalten wird im Folgenden verwendet. In der Definition werden allgemein Systeme und speziell künstliche Entitäten in die Beschreibung aufgenommen. Eine Entität ist dabei eine eindeutig identifizierbare Größe bzw. Einheit [24]. Weiterhin wird die Wechselwirkung mit sich selbst und mit der Umwelt, wobei zur Umwelt auch andere Systeme, Menschen (Organismen) und die unbelebte physische Welt gehören, hervorgehoben. Diese Betonung ist für das Konzept insofern wichtig, als dass die physischen Systeme sowohl mit der Umwelt wechselwirken, als auch dessen Digitalen Zwillingen. Diese Wechselwirkung kann entweder indirekt mittels der Verbindung mit dem Physischen Zwilling, als auch mit simulierten Umgebungselementen erfolgen. Außerdem wird verdeutlicht, dass sich das Verhalten als berechnete Reaktion auf Reize sonstiger Eingaben (*engl.:* Inputs) äußert. Die Eingaben müssen dabei nicht klar zuordenbar, sondern können intern/extern, bewusst/unbewusst, offen/verdeckt, freiwillig/unfreiwillig sein.

Aus der Definition nach Minton und Kahle [101] wird folgende Definition für die Verhaltenslogik technischer Systeme abgeleitet:

Die Verhaltenslogik eines technischen Systems gibt die Aktionen zur Erfüllung einer Aufgabe und die Reaktionen des Systems auf externe oder interne Stimuli unter Berücksichtigung des internen Zustandes und der Umwelt vor, zu der auch andere Systeme, Lebewesen und die unbelebte physische Umwelt gehören.

Im Folgenden werden die einzelnen Teile der abgeleiteten Definition näher erläutert und veranschaulicht. Die Definition wurde auf technische Systeme fokussiert und damit konkretisiert. Außerdem erfolgte eine Einschränkung auf die Logik des Systems. Unter der Logik werden dabei in Anlehnung an die Steuerungslogik einer Maschine nur die Aktionen und Reaktionen eines Systems verstanden, die durch die Programmierung und die Verschaltung der Elektronik vorbestimmt sind. Es wird z.B. nicht berücksichtigt, dass z.B. ein federndes Element bei einer äußeren Kraft eine Verformung und damit eine Änderung des äußeren Zustandes erfährt. Diese Verformung kann zwar in mechanischen Simulationen abgebildet werden, ist aber nicht die Folge der Programmierung oder der Elektronik. Wird dagegen die Verformung durch Sensorik vom System registriert, ist dies eine Reaktion der Verhaltenslogik auf einen externen Stimulus.

Unter Aktionen eines Systems zur Erfüllung einer Aufgaben werden internen und externen Zustandsänderungen verstanden, die vom System initiiert werden und aus der Programmierung entspringen sowie in der Elektronik verarbeitet werden. Ein CNC Produktionssystem z.B. kann die Aufgabe haben, auf Basis des G-Codes eine Bearbeitungsaufgabe

durchzuführen und wird als Aktionen die Achsen positionieren und im Falle eines Fräsers die Spindel antreiben. Eine andere Aufgabe eines Systems kann beispielsweise sein, einen internen Takt aufrechtzuerhalten und hoch zu zählen, um die aktuelle Betriebszeit zu erfassen. Diese Aktion zur Erfüllung einer Aufgabe ist zwar äußerlich nicht sichtbar und findet auf der Software- und Elektronikebene statt, gehört aber ebenso zur Verhaltenslogik eines technischen Systems.

Unter Reaktionen eines technischen Systems auf externe Stimuli wird Verhalten verstanden, welches durch die Außenwelt aufgebracht und von der Verhaltenslogik verarbeitet wird. Dazu gehört z.B. die Veränderung der Umwelt, die durch Sensorik erfasst wird und auf die durch die Programmierung und Verschaltung entsprechend reagiert wird. Besitzt ein Produktionssystem beispielsweise Lichtschranken zur Erfassung eines ungeplanten Eindringens in den Arbeitsbereich und dem Abschalten des Systems in Folge dessen, so ist dies eine Reaktion auf einen externen Stimulus. Wird das gleiche System durch einen Fernzugriff abgeschaltet, so ist dies ebenfalls ein externer Stimulus.

Zu internen Stimuli gehören dagegen Aktionen, die vom System selbst ausgelöst werden. Als Reaktion auf Überschreiten eines internen Grenzwertes kann das System Aktionen ausführen, wie beispielsweise das Einleiten eines Ruhemodus nach einer definierten Zeit ohne Aktivität. Falls der Digitale Zwilling beispielsweise auf Basis von prädiktiven Simulationen ein Ereignis auslöst, welches den internen oder externen Zustand des Systems verändert, so zählt es ebenfalls zu einem internen Stimulus.

Das Ziel des Konzepts ist die Integration der Verhaltenslogik eines Systems in den Digitalen Zwilling. Dadurch wird der Digitale Zwilling jedoch selbst zum Teil der Verhaltenslogik, weil es durch die bidirektionale Verbindung das interne und externe Verhalten mitbestimmt. Das Verhalten des technischen Systems wird durch den Digitalen Zwilling beeinflusst. Bei der Gesamtbetrachtung des technischen Systems und dem Digitalen Zwilling von einem *Zwillingssystem* gesprochen. Verwechslungen mit den Begriffen eines Zwillingssystems aus der Astronomie, der Werkstoffkunde oder aus der Domäne von Landschaftsmaschinen sind kontextbedingt nicht zu befürchten. Mit Verhaltenslogik eines technischen Systems ist demnach weiterhin nur der Physische Zwilling gemeint.

Weiter ist der Definition der Verhaltenslogik eines technischen Systems zu entnehmen, dass bei Aktivitäten und Reaktionen der interne Zustand berücksichtigt wird. Zum internen Zustand gehört dabei nicht nur der sichtbare Zustand des Systems, wie beispielsweise die aktuelle Position der Achsen eines Roboterarms, sondern z.B. auch die in der Software und Elektronik festgelegte Konfiguration. Die Berücksichtigung erfolgt demnach beispielsweise bei einer Überschreitung des Arbeitsbereichs. Dabei wird durch das System ermittelt, dass aufgrund der aktuellen Position und der konfigurierten Grenzen die Sollposition nicht angefahren werden kann. Als Reaktion wird die Bearbeitung verweigert bzw. eine vordefinierte, alternative Aktion durchführt.

Bei der Berücksichtigung der Umwelt werden sowohl die unbelebte physische Umwelt, als auch der Mensch und insbesondere andere Systeme betrachtet. Zur Erfassung der Umwelt benötigt das System Informationen zum aktuellen Zustand dieser Umwelt. Diese Informationen kann das System entweder über eigene Sensorik erfassen oder von externen Systemen, wie auch dem Digitalen Zwilling erhalten. Außerdem steht ein Cyber-Physisches System, welches als das technische Zielsystem des Konzepts betrachtet wird, in Verbindung mit anderen, teilweise global verteilten Systemen. Daraus wird abgeleitet, dass falls notwendig auch diese bei den Aktionen und Reaktionen des Systems berücksichtigt werden. Dies bedeutet insbesondere für den Digitalen Zwilling, dass die zu integrierende Verhaltenslogik nicht von etwaigen Verbindungen zu diesen Systemen abgetrennt werden darf, sondern weiterhin auch im Digitalen Zwilling bestehen muss, da es sonst nicht mehr der tatsächlichen Verhaltenslogik des technischen Systems entspricht. Wenn beispielsweise der Betrieb eines Produktionssystems davon abhängt, ob ein über ein Netzwerk verbundenes System beispielsweise einen Betriebsfreigabestatus aufrechterhält und falls dieser nicht vorliegt, der Betrieb eingestellt wird, so muss dies auch im Digitalen Zwilling repräsentiert werden. Als ein anschauliches Beispiel kann beispielsweise ein Lasersystem dienen, welches nur betrieben wird, solange Sicherheitsmechanismen, wie Türschalter und Lichtschranken, einen Freigabestatus aufrechterhalten, so muss diese Verhaltenslogik im Digitalen Zwilling ebenfalls abgebildet werden.

Die Definition der Verhaltenslogik und die ausführliche Erläuterung dient dazu, die Verhaltensabbildung im Digitalen Zwilling eindeutig zu bestimmen und einzugrenzen. Erst durch die vollständige Integration der nun definierten Verhaltenslogik kann die Verhaltensabbildung in Anwendungsfällen sinnvoll eingesetzt werden. Diese dabei helfen, das vorhersehbare und gewünschte Verhalten zu evaluieren, zu messen und zu bewerten, das vorhersehbare aber unerwünschte Verhalten zu eliminieren und unvorhersehbares und unerwünschtes Verhalten aufzudecken. Dies wird umso schwieriger, je komplexer ein System ist. Insbesondere vernetzte Systeme mit externer Verbindung können Zustände einnehmen und Verhalten aufweisen, welches bei der Entwicklung nicht vorhergesehen werden kann. Damit würde aber auch jegliche unvollständige Nachbildung des Verhaltens in einem Digitalen Zwilling schnell an Grenzen stoßen.

Für die menschenlesbare Darstellung des Verhaltens existieren unterschiedliche Modellierungssprachen und Abbildungsformen. Im Rahmen dieser Dissertation wird wie bereits bei der Beschreibung der Anwendungsfälle im Kapitel 3.3 auf Darstellungsformen nach UML 2 zurückgegriffen, wobei ein UML-Anwendungsfalldiagramm bereits eine Form der Verhaltensmodellierung ist.

Das Verhalten in Form von zeitlichen Abläufen nimmt in der Softwareentwicklung eine wichtige Rolle ein. Zur Modellierung des Verhaltens bietet UML 2 beschreibende Diagramme an, welche die internen Zustandsänderungen und das Zusammenwirken der

Einzelteile spezifizieren. Die sieben Verhaltensdiagramme sind das *Use-Case-Diagramm*, *Aktivitätsdiagramm*, *Zustandsautomat*, *Sequenzdiagramm*, *Kommunikationsdiagramm*, *Timingdiagramm* und *Interaktionsübersichtsdiagramm*. Aufgrund des Schwerpunktes auf der Interaktion, werden die vier letzteren Diagrammarten ab UML 2.5 als Interaktionsdiagramme bezeichnet. Die sieben Diagrammarten stellen jeweils andere Eigenschaften des Verhaltens eines Systems dar, sodass keines der Diagramme das vollständige Verhalten abbilden kann, sondern erst durch das Zusammenwirken mehrerer Diagramme.

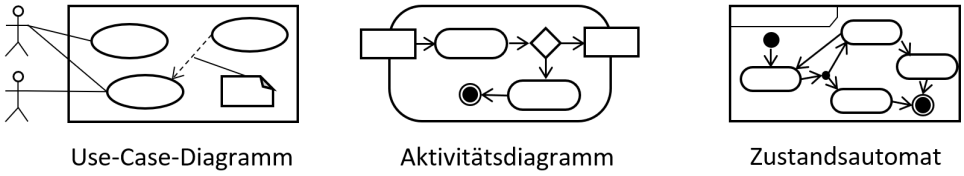

Abbildung 4.4: UML 2 Verhaltensdiagramme [in Anlehnung an [127]]

Die UML Verhaltensdiagramme umfassen im engeren Sinn das Use-Case-Diagramm, das Aktivitätsdiagramm und den Zustandsautomaten (siehe Abbildung 4.4). Im Use-Case-Diagramm wird beschrieben, in welchen konkreten Anwendungsfällen das System mit einem Anwender interagiert. Im Kapitel 3.3 beschreiben die Use-Case-Diagramme die anvisierte Verwendung des zu entwickelnden Systems und definiert damit insbesondere die Anforderungen an das Konzept. Das auf Basis des Konzepts entwickelte System kann jedoch von dem Wunschzustand abweichen, sodass die aufgestellten Use-Case-Diagramme keine Aussage über die tatsächliche Verhaltenslogik treffen.

Mittels der Aktivitätsdiagramme werden Aktivitäten eines Systems modelliert. Diese werden im Vorfeld zur Entwicklung der Software verwendet, sodass später die Algorithmen danach implementiert werden. Ein Aktivitätsdiagramm kann jedoch auch verwendet werden, um ein bereits in Software hinterlegtes Verhalten abzubilden, wobei aufgrund der höheren Abstraktion des Aktivitätsdiagramms das tatsächliche Verhalten nicht vollständig abgebildet wird, sondern nur als Annäherung gesehen werden muss. Aktivitätsdiagramme haben einen ablauforientierten Charakter, womit aktuelle Zustände nicht vollständig berücksichtigt werden können.

Eine alternative, zustandsorientierte Darstellung bieten die Zustandsautomaten. Diese beschreiben möglichen Zustände eines Systems und geben an, welche Ereignisse zu Zustandsübergängen führen. Die schematischen Diagramme aus Abbildung 4.5 zeigen verhaltensbeschreibende Modelle, die im Fokus auf die Darstellung der Interaktionen zwischen einzelnen Elementen spezialisiert sind. In einem Kommunikationsdiagramm werden die Zusammenhänge bei der Kommunikation zwischen einzelnen Kommunikati-

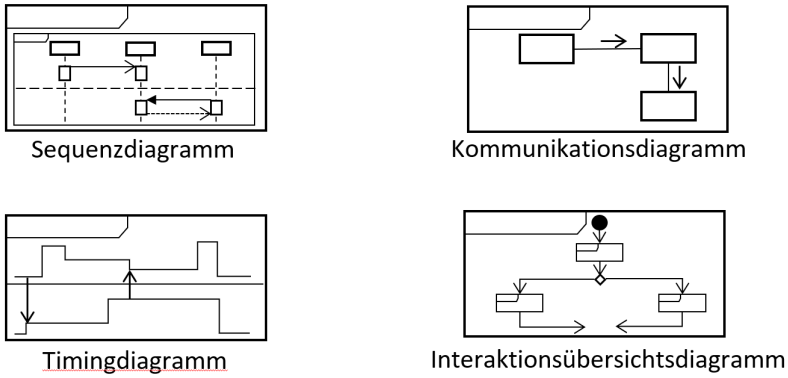

Abbildung 4.5: UML 2 Interaktionsdiagramme [in Anlehnung an [127]]

onspartnern bei der Verarbeitung einer Aufgabe modelliert.

In einem Sequenzdiagramm werden neben den Kommunikationspartnern auch die internen Abläufe visualisiert, wobei der Schwerpunkt dieses Diagramms auf der exakten Chronologie der einzelnen Aktivitäten liegt.

Die exakte Berücksichtigung der Zeit erfolgt in einem Timingdiagramm. Bei diesem Diagrammtyp bildet die Zeit eine Achse über welche die einzelnen Zustände der Systeme dargestellt werden. Hierbei kann insbesondere auch die Wechselwirkung zwischen einzelnen Interaktionspartnern und deren Einfluss auf die Zustände der Systeme dargestellt werden.

Ein Interaktionsübersichtsdiagramm bietet, wie der Name bereits impliziert, die Übersicht der Zusammenhänge einzelner Interaktionen. Mittels dieser Diagrammart lassen sich komplexe Interaktionen abbilden.

Im Rahmen der Dissertation werden die UML Interaktionsdiagramme ebenfalls benutzt, um bereits implementierte Verhaltenslogik abzubilden. Dabei kann sie jedoch nur als Näherung an das tatsächliche Verhalten betrachtet werden, da insbesondere bei komplexen Systemen, wie beispielsweise vernetzten Cyber-Physischen Systemen, nicht alle Eigenschaften der Interaktionen, an den mehrere Systeme beteiligt sein können, abbildbar sind.

Zusammenfassend wurde im vorigen Kapitel eine Definition für Verhaltenslogik technischer Systeme vorgestellt und veranschaulicht, welche Bedeutung diese auch für die Integration in den Digitalen Zwilling mit sich bringt. Damit wurde eine Basis für das nachfolgende Konzept geschaffen und auch die Möglichkeiten erläutert, das Verhalten hervorzuheben und zu visualisieren.

4.3 Virtualisierung der Verhaltenslogik

Der Gesamtprozess der Integration der Verhaltenslogik in den Digitalen Zwilling mittels einer Virtuellen Klemmleiste besteht demnach aus den zwei Teilprozessen *A1: Virtualisierung der Verhaltenslogik* und *A2: Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik*. Im folgenden Kapitel wird A1 näher erläutert und insbesondere beschrieben, wie die Schnittstelle zwischen der Elektronik und Software und dem restlichen System in Form einer *Virtuellen Klemmleiste* gestaltet werden kann.

Die Virtualisierung der Verhaltenslogik dient allgemein dem Zweck, diese im Digitalen Zwilling berücksichtigen zu können. Denn obgleich diese in Form von Software bereits digital vorliegt, kann sie nicht direkt verwendet werden, falls einerseits die Hardwareplattform des Physischen und Digitalen Zwillings sich unterscheiden und andererseits die Software-Schnittstelle im Physischen System nicht in der nativen Form übernommen werden kann, sondern an die Beschaffenheit des Digitalen Zwillings angepasst werden muss. Während beispielsweise im Physischen Zwilling an der physischen Elektronik ein Aktor mittels Kabel angeschlossen wird, wird der physische Aktor im Digitalen Zwilling durch ein virtuelles Modell des Aktors repräsentiert.

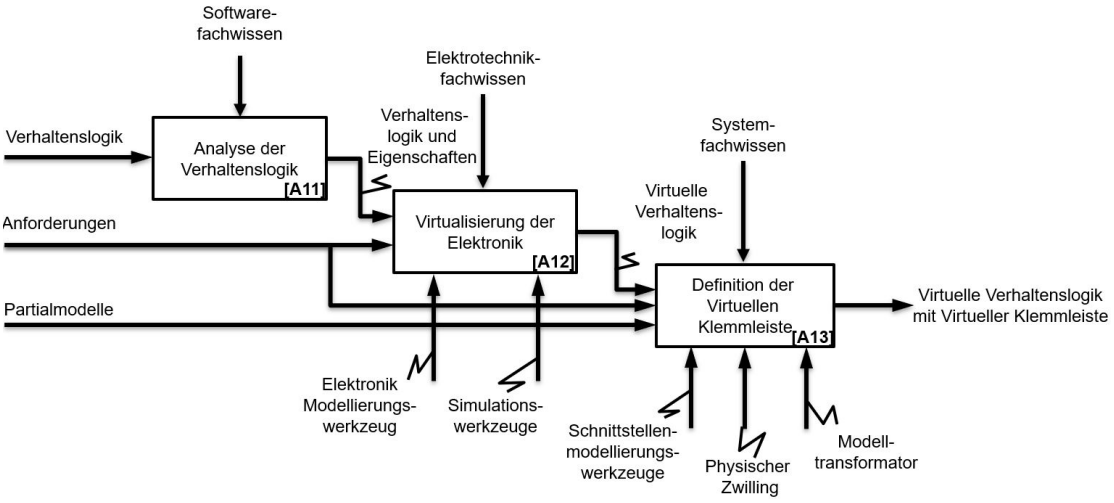

Abbildung 4.6: Teilprozess A1: Virtualisierung der Verhaltenslogik

Die Virtualisierung der Verhaltenslogik erfolgt in drei Teilprozessen, die in der Abbildung 4.6 mittels SADT dargestellt werden. In Teilprozess *A11: Analyse der Verhaltenslogik* wird die Verhaltenslogik des physischen Systems durch einen Softwareexperten hinsichtlich

der für die spätere Virtualisierung relevanten Eigenschaften untersucht. Die ermittelten Eigenschaften fließen zusammen mit der ansonsten unveränderten Verhaltenslogik in den Teilprozess *A12: Virtualisierung der Elektronik*. A12 dient dazu, mittels eines virtuellen Modells der Elektronik des betrachteten Systems, eine Plattform für die virtuelle Ausführung der Verhaltenslogik bereitzustellen. Hierfür verwendet ein Elektronikexperte die Elektronikmodelle aus der Entwicklung oder bildet diese mittels der Modellierungswerkzeuge für Elektronik nach. In beiden Fällen müssen Anforderungen berücksichtigt werden, die an die spätere Leistungsfähigkeit des Elektronikmodells gestellt werden, da diese beispielsweise grundsätzlich in unterschiedlichen Detaillierungsgraden abgebildet werden können. In einer Simulationsumgebung wird anschließend die modellierte Elektronik ausgeführt und dient als Basis für die eingebettete Software. Der Teilprozess A12 wird im Kapitel 4.3.2 näher beleuchtet. Im dritten Teilprozess *A13: Definition der Virtuellen Klemmleiste* wird auf Basis der Verhaltenslogik und deren Eigenschaften eine Virtuelle Klemmleiste definiert und modelliert. Die Virtuelle Klemmleiste dient dazu, eine definierte Schnittstelle zwischen der virtuell ausgeführten Verhaltenslogik und den virtuellen Abbildungen des physischen Teile des Physischen Zwillings zu erschaffen. Diese Aufgabe wird von einem Systemexperten übernommen, der einerseits die Verhaltenslogik verstehen und anderseits den Physischen Zwilling analytisch auslegen und modellieren kann. Eine ausführliche Beschreibung von A13 erfolgt im Kapitel 4.3.3.

4.3.1 Analyse der Verhaltenslogik

Im Teilprozess *A11* der Virtualisierung der Verhaltenslogik wird zunächst eine gezielte Analyse der für die Virtualisierung benötigten Eigenschaften der eingebetteten Software vorgenommen. Der Teilprozess dient insbesondere dazu, die Rahmenbedingungen zu ermitteln, unter denen die eingebettete Software im Digitalen Zwilling verwendet werden kann. Bei der Analyse der eingebetteten Software sind weniger die internen Abläufe interessant, als mehr die äußere Wirkung. Das Ziel ist es dabei, die Funktion der Software nicht zu verändern, sondern diese möglichst analog zur Wirkungsweise im Physischen Zwilling zu virtualisieren. So ist die Ausführungsgeschwindigkeit beispielsweise eine wichtige Eigenschaft, die bei der Virtualisierung berücksichtigt werden muss. Ein Digitaler Zwilling, der mit einem Bruchteil der Geschwindigkeit des realen Systems betrieben werden kann, unterscheidet sich demnach wesentlich hinsichtlich des Verhaltens vom Physischen Zwilling.

Die Analyse der eingebetteten Software hängt maßgeblich vom betrachteten System ab und erfordert Softwarefachwissen, um die wesentlichen Eigenschaften abzuleiten. Bei der Analyse wird einerseits die Software betrachtet und andererseits bereits die Hardwareabhängigkeit untersucht. Bei der Softwareanalyse muss unter anderem erfasst werden,

in welcher Sprache oder auch unterschiedlichen Sprachen die Software implementiert wurde. Es muss analysiert werden, welche Softwareteile enthalten sind und wie diese miteinander interagieren. Neben monolithischem Aufbau sind auch modularisierte Architekturen möglich, die zusätzlich Parallelität in der Ausführung aufweisen können und berücksichtigt werden müssen. Die einzelnen Softwaremodule besitzen gemeinsame Schnittstellen, die im Digitalen Zwilling ebenfalls aufgegriffen werden müssen. Bei Cyber-Physischen Systemen ist die Vernetzung eine zentrale Eigenschaft des Systems, die auch in der Software repräsentiert ist und dazu führt, dass das System in der Funktion nicht abgeschlossen betrachtet werden kann. Jedoch muss eine klare Systemgrenze gezogen werden, welche die Funktionen eingrenzt, die im Digitalen Zwilling abgebildet werden sollen und gleichzeitig die Schnittstellen festlegt, über welche die externen Ressourcen eingebunden werden müssen. Die Kommunikation mit externen Ressourcen kann dabei über unterschiedliche Kommunikationskanäle und Protokolle stattfinden. Für die externen Schnittstellen müssen zudem die Ein- und Ausgaben bekannt sein, um in späteren Prozessschritten diese Datenflüsse abbilden zu können. In der Software können unterschiedliche Arbeitstaktraten zum Einsatz kommen, indem beispielsweise die interne Verarbeitung in einer festgelegten, vom Mikrocontroller abhängigen Taktrate, stattfindet. Die Datenübertragung über eine serielle Schnittstelle kann jedoch in einer abweichenden Taktrate (Baudrate) konfiguriert, die einem festgelegten Verhältnis zum Arbeitstakt steht. Diese und weitere Eigenschaften haben Einfluss auf die Virtualisierung der Verhaltenslogik und müssen in einer Softwareanalyse ermittelt werden. Die Analyse kann dabei mittels der Quellsoftware und den Kommentaren in der Quellsoftware oder auf Basis von technischer Dokumentation durchgeführt werden. Im ersten Fall wird dabei die Annahme getroffen, dass die Software als Quellcode der eingebetteten Software vorliegt. Die Prämisse wird in der aktuellen Praxis oft nicht erfüllt, da der Urheber der Software damit das inhärente Know-How offenlegt. Obgleich für das Reengineering von bereits kompilierter Software ebenfalls Methoden und Werkzeuge existieren, müssen die Vertragsbedingungen des Eigentümers der Software berücksichtigt werden. Die Analyse der technischen Dokumentation erfordert dagegen keinen Quellcode, sondern setzt eine vollständige Dokumentation voraus, die durch Rücksprache mit dem Urheber der Software ergänzt werden kann. Eine Voraussetzung hierbei ist jedoch das Vorhandensein des Maschinencodes in einem Format, welcher mit der Hardwaresimulation kompatibel ist.

Durch die hardwaregebundene Eigenschaft eingebetteter Software, können aus dem Quellcode oder der Dokumentation bereits Rückschlüsse auf die verwendete Hardware erfasst werden. Zu diesen Informationen gehören die grundlegende Hardwarearchitektur und die verwendeten Hardwareelementen wie der Mikroprozessor, der Random Access Memory (RAM), der Read Only Memory (ROM), die verwendeten Kommunikationsschnittstellen und Hardwarekomponenten. Weiterhin bedeutsam sind die durch die Software

verwendeten Register, die über Datentransfersysteme, wie z.B. Kommunikationsbussystem und Eingaben/Ausgaben adressiert werden. Die Informationen zur Hardware und den Schnittstellen werden in späteren Prozessschritten benötigt, um die Elektroniksimulation und die Virtuelle Klemmleiste auszulegen. In einigen Fällen werden nicht alle Hardwareelemente und Funktionen tatsächlich durch die Software verwendet, wodurch diese beispielsweise im Digitalen Zwilling vernachlässigt werden können, um die benötigten Rechenressourcen zu reduzieren und die Simulationszeit zu beschleunigen.

Zusammenfassend werden im Prozessschritt *A11: Verhaltenslogik analysieren* die Informationen extrahiert, welche für die Umsetzung der Simulation und die Definition der *Virtuellen Klemmleiste* benötigt werden. Dazu gehören die:

- verwendeten Elektronikkomponenten (wie z.B. Mikrocontroller, Speicher, Wandler, Verstärker),
- konfigurierte Taktrate des Mikroprozessors,
- Kommunikationsschnittstellen mit Angabe der Kommunikationsprotokolle, Baudraten sowie der verwendeten Ports und
- eine Liste der Parameter und Variablen mit Angabe der zugeordneten Ein- und Ausgabeports, die während der Ausführung gelesen bzw. geschrieben werden.

Diese Informationen werden gemeinsam mit der Verhaltenslogik an den nächsten Prozessschritt übergeben.

4.3.2 Virtualisierung der Elektronik

Die technische Grundlage für die Virtualisierung der Verhaltenslogik stellt ein ebenfalls virtuelles, simulationsfähiges Modell der Elektronik, in welchem die eingebettete Software auf einem Computer ausgeführt wird, dar. Wie im Kapitel 2.4.3 beschrieben, ist dies erforderlich, da die eingebettete Software aufgrund abweichender Hardwarearchitektur nativ nicht ausgeführt werden kann. In der Abbildung 4.7 ist der Teilprozess *A12: Virtualisierung der Elektronik* in drei Schritte unterteilt, die im Folgenden näher erläutert werden.

Im ersten Teilprozess A121 wird ein generisches Modell der eingesetzten Elektronik erstellt. Dies ist eine domänenspezifische Aufgabe, die ein tiefgreifendes Elektronikfachwissen im Verständnis, Umgang, Entwicklung und Modellierung von elektronischen Verschaltungen in allen Prozessschritten erfordert. Der in der Abbildung 4.7 dargestellte Teilprozess beginnt mit der Modellierung der Elektronik, die im Physischen Zwilling eingesetzt wird. Dabei muss diese nicht zwangsweise grundlegend neu erstellt werden,

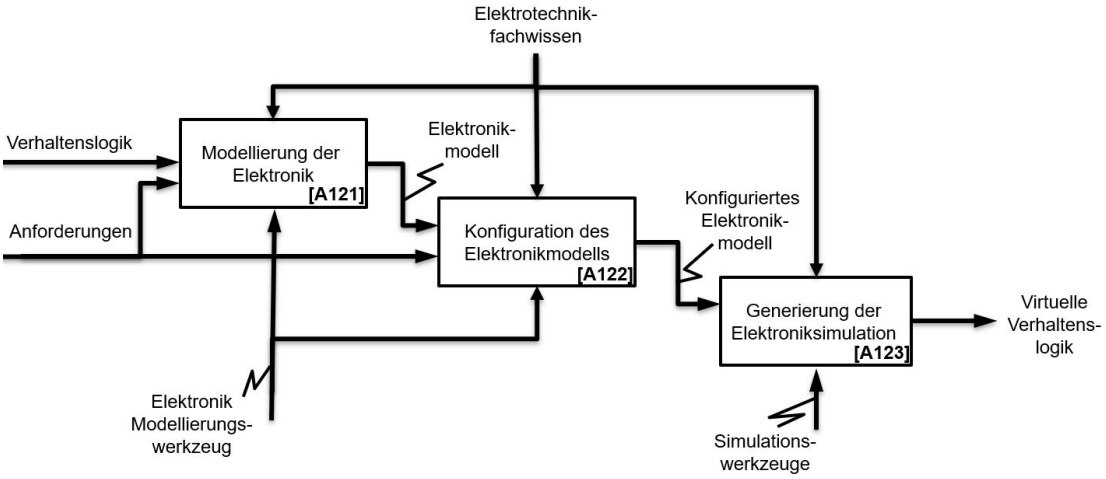

Abbildung 4.7: Teilprozess A12: Virtualisierung der Elektronik

sondern es können Vorlagen oder auch das vollständige Modell aus der ursprünglichen Entwicklung herangezogen und erweitert werden. Der Umfang dieser Aufgabe hängt also maßgeblich davon ab, ob bereits Modelle und Know-How zur Verfügung stehen, die verwendet und weiterentwickelt werden. Somit kann der Arbeitsaufwand, den Teilprozess *A121* durchzuführen stark variieren. Vor und während der Modellierung der Elektronik müssen Fragestellungen in Abhängigkeit von der Verhaltenslogik und den Anforderungen beantwortet werden, die im Folgenden erläutert werden. Vor der eigentlichen Modellierung muss eine geeignete Plattform bzw. Modellierungswerkzeug ausgewählt werden. Die Auswahl hängt dabei von Faktoren ab wie z.B. der Architektur der zu modellierenden Elektronik und insbesondere des Mikrocontrollers, der Programmiersprache, der grundlegenden Solver, der zur Verfügung stehenden Datenbanken für elektronische Modelle, der Erweiterbarkeit des Modellierungswerkzeugs und der vorhandenen APIs. Dabei müssen neben der Verhaltenslogik des Physischen Zwillings auch die Anforderungen berücksichtigt werden. Falls beispielsweise eine Simulation in naher Echtzeit oder sogar eine vorauseilende Simulation erwünscht ist, muss dies in der Auswahl berücksichtigt werden. Zum aktuellen Stand der Technik existieren tatsächlich nur wenige Werkzeuge, die in Betracht gezogen werden können und keines, was für die im Konzept vorgesehene Funktion zielgerichtet ausgelegt ist. Für die unterschiedlichen Hersteller von Mikroprozessoren stehen dabei wenige bis keine Werkzeuge bereit, die zudem untereinander nicht kompatibel sind. Ist eine geeignete Modellierungsumgebung identifiziert, muss beispielsweise weiter

eruiert werden, inwiefern neben dem Mikroprozessor noch weitere Elektronikmodelle, wie Motortreiber, Signalumwandler oder Kommunikationsschnittstellen bereits unterstützt werden bzw. bereits als Modell vorliegen. Falls erforderliche Komponentenmodelle fehlen, müssen diese eigenständig generiert werden. Kommerzielle Modellierungswerkzeuge binden externe Datenbanken ein, welche die Teilmodelle ggf. anbieten. Dabei unterscheidet sich der Umfang der angebotenen Funktionalität des Modells von einer einfachen visuellen Darstellung bis zur vollständigen Beschreibung der Funktionsweise mittels mathematischer Gleichungen. Diese Fragestellungen gehören zu den Aufgaben des Elektronikexperten während der Modellierung und sind ebenfalls abhängig von gegebenen Randbedingungen. Eine hervorzuhebende Anforderung, welche die Voraussetzung für die spätere Erstellung der *Virtuellen Klemmleiste* und die Kopplung mit Partialmodellen des Digitalen Zwillings, ist die Verfügbarkeit geeigneter Schnittstellen, um die Datenflüsse aus- und in die Simulation leiten zu können.

Anschließend an die Modellierung der Elektronik wird der Teilprozess *A122: Konfiguration des Elektronikmodells* durchgeführt. In diesem Teilprozess muss das noch generische Modell der eingebetteten Elektronik für den Einsatz im Digitalen Zwilling konfiguriert werden. Die konkrete Konfiguration hängt von verwendeter Entwicklungsumgebung ab. Grundsätzlich muss jedoch sichergestellt werden, dass die anschließende Simulation der eingebetteten Software den Anforderungen und dem Einsatz im Digitalen Zwilling entspricht. Es muss definiert werden, welche Hardwaremodule tatsächlich simuliert werden sollen. In Abhängigkeit vom Anwendungsfall und den daraus resultierenden Anforderungen kann ein höherer oder niedrigerer Detaillierungsgrad adressiert werden. Eine umfangreichere Abbildung der eingebetteten Elektronik würde beispielsweise mehr Simulationsressourcen in Anspruch nehmen und das Verhältnis zwischen der Simulationszeit und der simulierten Zeit ungünstig beeinflussen. Aus diesem Grund können Abstraktionen notwendig werden, welche zwar die Granularität und den Detaillierungsgrad erniedrigen, jedoch die Simulation beschleunigen. Für diesen Zweck können beispielsweise vereinfachte Modelle elektronischer Komponenten eingesetzt werden bzw. ein geeigneter Detaillierungsgrad ausgewählt werden. Die sinkende Genauigkeit kann dabei vorzugsweise auf Funktionen eingegrenzt werden, die in der Anwendung des Digitalen Zwillings eine untergeordnete bis keine Bedeutung haben. So kann beispielsweise für einen Signalumwandler angenommen werden, dass dieser eine stets exakte Umwandlung durchführen kann. Dadurch kann statt einer komplexen, auf einem Solver basierenden Abbildung der Funktionsweise ein vereinfachter Zusammenhang zwischen dem Eingang und dem Ausgang des Signalumwandlers gewählt werden. Komponenten, welche durch die eingebettete Software nicht verwendet werden, können zudem im Elektronikmodell inaktiv gesetzt oder entfernt werden, damit diese während der Simulation nicht berücksichtigt werden. Die getroffenen Maßnahmen dienen zusammenfassend dazu, einen anforderungsgerech-

ten Ausgleich zwischen der Simulationsgeschwindigkeit und dem Detaillierungsgrad zu finden. Nach erfolgter Auswahl der einzelnen Elektronikmodelle werden die verwendeten Schnittstellen konfiguriert sowie die Takt- und Baudraten gemäß den Angaben aus dem vorigen Teilprozess eingestellt.

Das generierte oder aus der Produktentwicklung wiederverwendete Simulationsmodell der Elektronik ist nach der Konfiguration noch nicht bereit, im Digitalen Zwilling eingesetzt zu werden, sondern muss zunächst in einer konformen Simulationsumgebung integriert werden. Die Simulationsumgebung muss generelle Anforderungen erfüllen, die aus dem Kontext des Digitalen Zwilling abgeleitet werden und im Kapitel 3.4 erläutert sind. Ein grundlegender Unterschied zwischen einer Simulation in der Produktentwicklung und einer Simulation im Rahmen eines Digitalen Zwillings besteht darin, dass die letztere über einen längeren Zeitraum ausgeführt wird und in einer bidirektionalen Verbindung mit anderen Simulationen und sonstigen Datenquellen steht, anstatt auf Basis eines festen Inputdecks die Simulation einmalig durchzuführen. Auch wenn in einer Simulation im klassischen Sinne mehrere Inputdecks automatisiert verarbeitet werden können, so werden diese Berechnungen unter starren Bedingungen nacheinander durchgeführt. In einem Digitalen Zwilling werden die Daten nicht in einem Inputdeck kontrolliert zusammengestellt, sondern werden im laufenden Betrieb des Systems erzeugt und sehr zeitnah weiterverarbeitet. Die Simulation muss dabei mit anderen Datenquellen und -empfängern verknüpft werden. Dies erfordert neben geeigneten Schnittstellen und gemeinsamen Datenformaten auch die Überwindung etwaiger Lücken zwischen den Simulationen, die dadurch entstehen können, dass die Ausgabe einer Simulation nicht oder nicht vollständig der Eingabe einer darauffolgenden Simulation entspricht. In diesen Fällen müssen individuelle Lösungen erarbeitet werden, um die zunächst inkompatible Simulationen zu harmonisieren. Diese und weitere Überlegungen müssen im nächsten Teilprozess *A123: Generierung der Elektroniksimulation* berücksichtigt werden. Das Ziel dieses Teilprozesses ist es, auf Basis der Elektronikmodells und der eingebetteten Software eine ausführbare Elektroniksimulation zu erschaffen, die für einen flexiblen Einsatz im Digitalen Zwilling bereitsteht. Diese muss sowohl Daten von externen Quellen, die über unterschiedliche Schnittstellen kontinuierlich übermittelt werden können, empfangen und selbsttätig verarbeiten, als auch die Simulationsergebnisse in einem permanenten Datenstrom und einer ausreichenden Geschwindigkeit ausgeben können. Die erforderliche Flexibilität muss zudem beinhalten, dass die einzelnen Simulationselemente auf verteilten Berechnungsressourcen lokalisiert sein können. In diesem Zusammenhang kann es auch erforderlich sein, die Schnittstellen auf standardisierten und offenen Formaten aufzubauen, da proprietäre Technologien durch den eher geschlossenen Charakter für den interdisziplinären Kontext weniger geeignet sind. Dadurch kann es notwendig sein, die Elektronikmodelle aus den domänenspezifischen Entwicklungs- und Simulationsumgebungen mittels geeig-

neter Technologien und Formate zu exportieren. In dem vorgestellten Teilprozess *A123* wird sowohl ein tiefes Verständnis der Elektronikmodelle und der Simulationen, als auch kontextspezifisches Know-How zum Digitalen Zwilling benötigt, um eine letztlich eine Elektroniksimulation bereitstellen zu können.

4.3.3 Definition der Virtuellen Klemmleiste

Im Teilprozess *A13: Definition der Virtuellen Klemmleiste* erfolgt die Bildung einer Schnittstelle zwischen der eingebetteten Software und Hardware und den Systemsimulationen. Der in der Abbildung 4.8 dargestellte Teilprozess besteht aus drei wiederum untergeordneten Prozessen: *A131: Analyse des Physischen Zwillings*, *A132: Analyse der Partialmodelle* und *A133: Generierung der Virtuellen Klemmleiste*. Die wesentliche Aufgabe der ersten beiden Teilprozesse besteht darin, die für einen Anwendungsfall notwendigen Parameter und Variablen[1] aus dem Physischen Zwilling und den Partialmodellen, welche definierte Eigenschaften des physischen Systems abbilden, zu ermitteln und diese mit den Parametern und Variablen aus der eingebetteten Software und Elektronik zusammenzuführen. In diesem Teilprozess wird somit die Brücke zwischen unterschiedlichen Teilen des Digitalen Zwillings in Form einer *Virtuellen Klemmleiste* gebildet.

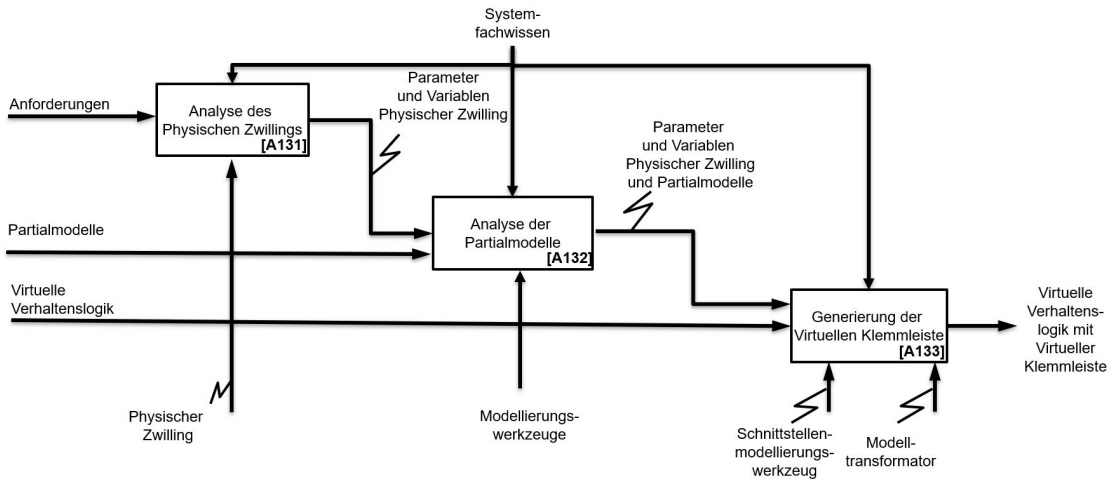

Abbildung 4.8: Teilprozess A13: Definition der Virtuellen Klemmleiste

Im ersten Schritt des Teilprozesses A13 erfolgt die Analyse des Physischen Zwillings.

[1] Im Rahmen der Dissertation werden unter *Parametern* die zur Laufzeit des Digitalen Zwillings unveränderliche Größen verstanden, während *Variablen* sich zur Laufzeit verändernde Größen bezeichnen

Dabei fließen als Eingabe die Anforderungen an das gesamte Zwillingssystem in die Analyse mit ein, weil daraus hervorgeht, welche Eigenschaften des Systems betrachtet werden sollen. Der Fokus der Analyse der Hardware kann beispielsweise auf kinematischen Zusammenhängen eines Systems liegen. In diesem Fall müssen Komponenten betrachtet werden, die für das kinematische Verhalten maßgeblich sind. Im vorgestellten Beispiel kann es dabei notwendig werden, sowohl das virtuelle als auch das physische System mit zusätzlicher Sensorik auszustatten, um die durch die Anforderungen vorgegebenen Analysen des Verhaltens überwachen zu können.

Im Teilprozess A13 ist es somit die Aufgabe eines Systemingenieurs, auf Grundlage der Anforderungen das physische System zu analysieren und die Parameter und Variablen zu ermitteln, welche für die geforderte Abbildung des Verhaltens im Digitalen Zwilling notwendig sind. Bei der Analyse ist es zielführend, mithilfe der technischen Dokumentation des Systems und ausgehend von der Steuerungselektronik die angeschlossenen Aktoren und Sensoren mittels Hersteller und Typenbezeichnung zu identifizieren. Anschließend werden die Schnittstellen der Aktoren und Sensoren ermittelt. Die Schnittstellen dieser Komponenten hängen von der Art der Komponenten ab. Aktoren müssen z.B. durch definierte Signalarten angesteuert werden, während Sensoren Daten in Kodierungen erzeugen und über Schnittstellen übertragen. Aktoren sind mit einer Leistungselektronik verbunden, die aus einem Steuersignal die aktorspezifischen Spannungen/Ströme generiert. Sensoren sind ebenfalls mit sensorindividueller Elektronik verbunden, welche aus den sensorspezifischen Spannungen/Strömen entsprechende Datensignale erzeugt. Für die spätere Generierung der Virtuellen Klemmleiste werden lediglich die jeweils verwendeten Abschlüsse (Ports) der einzelnen Aktoren und Sensoren an die Steuerungselektronik benötigt. Die Abbildung der Aktoren und Sensoren sowie der zugehörigen Leistungselektronik erfolgt in den entsprechenden Partialmodellen des Digitalen Zwillings. Die identifizierten Aktoren und Sensoren sowie die zugehörigen Ein- und Ausgänge werden als Ausgabe des Teilprozesses A131 an die nachfolgende Aktivität übergeben.

Der nachfolgende Teilprozess *A132: Analyse der Partialmodelle* hat zum Ziel, die Partialmodelle, welche die Komponenten und Eigenschaften des physischen Systems virtuell Abbilden und als Eingabe im Teilprozess betrachtet werden, zu analysieren und auszuwählen. Dies ist erforderlich, um auf Basis der *Virtuellen Klemmleiste* eine Verbindung zwischen dem Physischen und Digitalen Zwilling bilden zu können. Insbesondere muss der Systemingenieur untersuchen, ob Unterschiede in der Schnittstelle der realen Komponenten und den zugehörigen Partialmodellen bestehen. Diese Informationen sind erforderlich, da es das Ziel des Gesamtvorhabens ist, die identische eingebettete Software zu verwenden, um sowohl den Physischen als auch den Digitalen Zwilling steuern zu können. Da dabei die eingebettete Software selbst nicht verändert wird, müssen die Partialmodelle entsprechend ausgewählt werden, um in Analogie zu den physischen Komponenten gesteuert werden zu

können. Es könnte erforderlich sein, Anpassungen in den Partialmodellinstanzen durchzuführen, um einerseits im Kontext eines Digitalen Zwillings eingesetzt zu werden und andererseits, um die Daten der eingebetteten Software verarbeiten zu können. Dabei werden Modellierungswerkzeuge eingesetzt, welche zur Bearbeitung der individuellen Partialmodelle notwendig sind. Die möglichen Anpassungen und Erstellungen der Partialmodelle von Aktoren und Sensoren liegen außerhalb des Betrachtungsbereichs dieser Dissertation, sondern es wird davon ausgegangen, dass geeignete Partialmodelle z.B. von den Herstellern der Aktoren und Sensoren bereitgestellt und vom Systemingenieur ausgewählt werden. Nachdem der Systemingenieur die Partialmodelle der Aktoren und Sensoren ausgewählt hat, erfolgt der nächste Prozessschritt.

Nach der Extraktion der notwendigen Parameter, die für die Kommunikation zwischen den einzelnen Elementen des Zwillingssystems benötigt werden, erfolgt der *Teilprozess A133 Generierung der Virtuellen Klemmleiste*. Neben der Eingabe der Parameter und Variablen aus dem Physischen Zwilling und den Partialmodellen, fließt auch die eingebettete Software und die Elektronik mit in die Generierung der *Virtuellen Klemmleiste* als finalen Prozessschritt mit ein. Gemäß der Abbildung 4.9 ist das Ziel des Teilprozesses A133 die Ausprägung der einzelnen *Virtuellen Klemmen*. An diesen Klemmen werden im nachfolgenden Prozess der Erweiterung des Digitalen Zwilling durch virtualisierte Verhaltenslogik die einzelnen Partialmodelle des Digitalen Zwillings eingebunden.

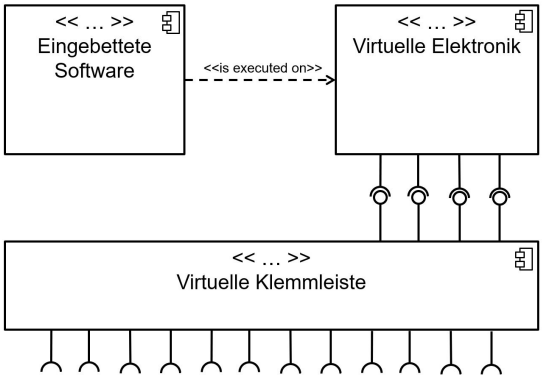

Abbildung 4.9: Virtuelle Klemmleiste als Schnittstelle zur virtuellen Elektronik. An die Virtuellen Klemmen können weitere Partialmodelle des Digitalen Zwillings angebunden werden.

Das Konzept der *Virtuellen Klemmleiste* (Abbildung 4.10) orientiert sich von der Grundidee an einer physischen Klemmleiste, welche insbesondere in Produktionssystemen zum Einsatz kommt, um die Komponenten eines Systems auf eine standardisierte Art

miteinander zu verbinden.

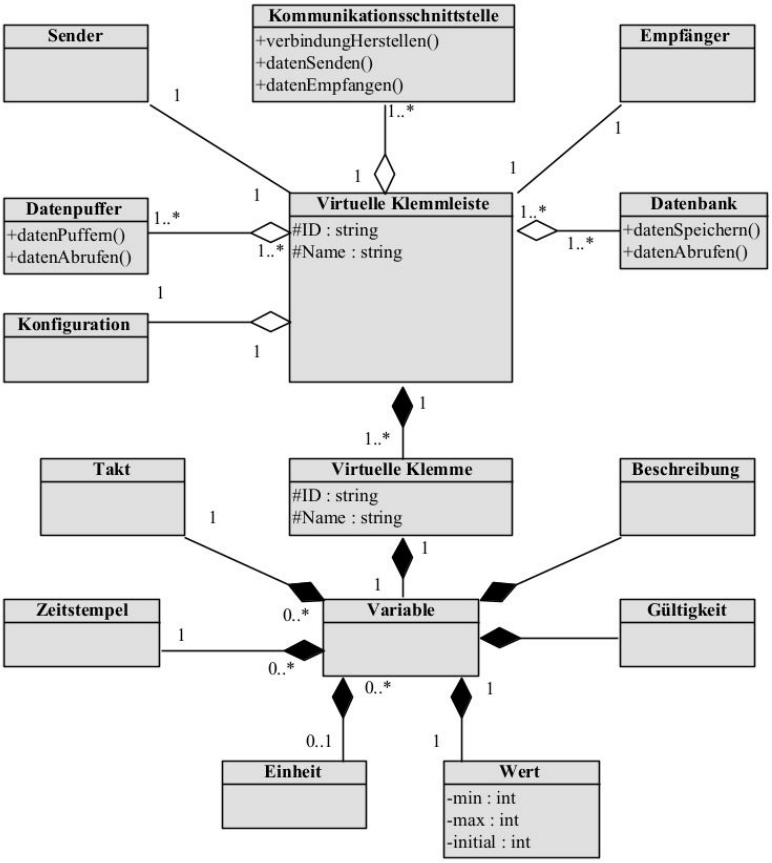

Abbildung 4.10: Modell der Virtuellen Klemmleiste

Die Virtuelle Klemmleiste hat eine eindeutige *ID*, worüber sie mittels einer Kommunikationsschnittstelle von unterschiedlichen Partialmodelle angebunden werden kann. Je nach Anwendungsfall treten die einzelnen Partialmodelle als *Sender* oder *Empfänger* auf, wobei die Rollen auch während einer Kommunikation wechseln können. Die Kommunikationsschnittstelle ermöglicht es, eine Verbindung herzustellen [+*verbindungHerstellen()*], Daten zu senden [+*datenSenden()*] und zu empfangen [+*datenEmpfangen()*]. Die einzelnen Kommunikationskanäle der *Virtuellen Klemmleiste* werden über einzelnen *Virtuellen Klemmen* adressiert. Über die Virtuellen Klemmen werden Daten ausgetauscht, die eine feste Zuweisung haben und in der Regel über die Zeit veränderlich sind - weshalb sie in der

Abbildung 4.10 als *Variable* bezeichnet werden und auch eine menschenlesbare Beschreibung besitzen. Dementsprechend kann beispielsweise ein Sensorwert eines Systems an einer vordefinierten Virtuellen Klemme bereitgestellt werden. Jedes Datum wird in einem vordefinierten Takt aktualisiert. Der *Wert* der Variable hat neben einem Initialwert, noch einen minimalen und maximalen Grenzwert. Die Grenzwerte dienen dazu, stets einen Gültigkeitsbereich für den Wert einer Variablen festlegen zu können. Ebenso wird die Gültigkeitsdauer eines Werts mittels einer zeitlichen *Gültigkeit* festgelegt. Der Zeitpunkt der Bereitstellung ist dabei mit einem *Zeitstempel* gekennzeichnet, sodass stets auch eine Reihenfolge der Variablen festgelegt werden kann. Eine weitere bedeutsame Eigenschaft jeder Variable ist die *Einheit*, die sowohl im internationalen Einheitensystem, als auch in abgeleiteter Form vorliegen kann. Die Einheit kann insbesondere genutzt werden, um die Kompatibilität von miteinander gekoppelten Partialmodellen zu analysieren.

Die Virtuelle Klemmleiste besitzt eine *Konfiguration*, worüber die wesentlichen Einstellungen festgelegt werden. Weiterhin ist es erforderlich, einen *Datenpuffer* zu integrieren, um beispielsweise Unterschiede in der Übertragungsgeschwindigkeit zwischen Sender und Empfängen in einem kurzen Zeitraum ausgleichen zu können. Für langfristige Speicherung von Daten steht eine *Datenbank* mit den Hauptfunktionen [+*datenSpeichern()*] und [+*datenAbrufen()*], bereit.

Eine konzeptionell-analytische Natur der Beschreibung der Schnittstelle reicht nicht aus, um diese im Betrieb verwenden zu können. Es wird letztlich eine ausführbare Software benötigt, die in einer *Server-Client* bzw. auch einer *Publish-Subscribe* Architektur eingesetzt werden kann.

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<virtuelleKlemmleiste name="..." id="...">
    <virtuelleKlemme name="..." id="...">
        <beschreibung></beschreibung>
        <takt></takt>
        <einheit></einheit>
        <gueltigkeit></gueltigkeit>
        <initialWert></initialWert>
        <min></min>
        <max></max>
    </virtuelleKlemme>
</virtuelleKlemmleiste>
```

Abbildung 4.11: Modellierung einer Virtuellen Klemmleiste in XML

Eine technologieunabhängige Beschreibung der Virtuelle Klemmleiste mithilfe der Auszeichnungssprache XML ist exemplarisch in der Abbildung 4.11 dargestellt. Neben

der XML-Deklaration zu Beginn der Datei wird anschließend die Virtuelle Klemmleiste definiert. Im Start-Tag des *virtuelleKlemmleiste*-Elements werden gemäß dem Modell aus Abbildung 4.10 die Attribute Identifikation (*id*) und Name (*name*) angegeben. Innerhalb des *virtuelleKlemmleiste*-Elements werden die einzelnen Virtuellen Klemmen definiert. Zu der Spezifikation einer Virtuellen Klemme gehört eine Beschreibung, ein Takt, eine Einheit, eine Gültigkeit, ein Initialwert sowie die Angabe eines minimalen und maximalen Grenzwertes. Die vorgestellte Beschreibung ist proprietär und könnte nur mittels darauf ausgelegte Softwareimplementierungen verwendet bzw. durch spezialisierte XML-Parser importiert werden. Zudem würde ein XML Schema zur Beschreibung der Grammatik und für eine Validierung der Schnittstellenbeschreibung benötigt werden. Aus diesen Gründen ist eine geeignete und bereits verbreitete Technologie vorzuziehen, weshalb zur Umsetzung einer ausführbaren Schnittstelle die OPC UA Technologie verwendet wird. Eine einführende Beschreibung von OPC UA kann Kapitel 2.4.2 entnommen werden.

Mithilfe der Knoten und Referenzen von OPC UA kann die Virtuelle Klemmleiste in einem maschinenlesbaren Format modelliert werden. Die Typenbeschreibungen werden dabei genutzt, um die Referenzen-, Objekt-, Variablen- und Datentypen der Schnittstellenbeschreibung zu definieren. Mit dem Ziel der branchenweiten Standardisierung werden die Typen durch Organisationen und Verbände erarbeitet und in Basismodellen oder den Companion Specifications genormt. Zum aktuellen Stand existiert keine Companion Specification, die bereits eine Virtuelle Klemmleiste beinhaltet, aus diesem Grund wird im Folgenden konzeptionell beschrieben, wie die Virtuelle Klemmleiste mittels OPC UA umzusetzen ist. Dies ergibt eine weiterhin proprietäre Lösung, die jedoch aufgrund der gleichen Technologiebasis als individuelle Erweiterung in ein bestehendes OPC UA Informationsmodelle aufgenommen werden kann. Die Modellierung erfolgt mittels eines OPC UA Modellierungswerkzeugs[2]. In der Abbildung 4.12 ist ein Ausschnitt aus einem OPC UA Informationsmodell der Virtuellen Klemmleiste für das Beispielsystem *Laserplotter* dargestellt. In dem abgebildetem XML-Ausschnitt wird ein Objekt Laserplotter definiert (Zeile 10-18), welches eine Referenz auf die Virtuelle Klemmleiste unter der ID *i=5005* aufweist. Im Objekt der Virtuellen Klemmleiste (Zeile 19-25) existiert wiederum ein Verweis auf alle Virtuellen Klemmen, wie z.B. eine Virtuelle Klemme mit dem Namen *VK_X* und der ID *i=6012*.

OPC UA bietet alle Funktionen, um die Virtuelle Klemmleiste abzubilden. Die Gültigkeit einer Variablen kann mit dem Attribut *MinimumSamplingInterval* angegeben werden. Die Einheit der Variable wird mit der Variableneigenschaft (*engl.: Standard Property*) *EngineeringUnits* spezifiziert.

[2] geeignete Modellierungswerkzeuge sind z.B.: Unified Automation uaModeller, Free OPC UA Modeller, OPC Foundation UA Model Compiler

```xml
<?xml version="1.0" encoding="utf-8"?>
<UANodeSet
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:uax="http://opcfoundation.org/UA/2008/02/Types.xsd"
    ...
    <NamespaceUris>
        <Uri>http://DiK.org/Laserplotter/</Uri>
    </NamespaceUris>
    ...
    <UAObject NodeId="ns=1;i=5006" BrowseName="1:Laserplotter">
        <DisplayName>Laserplotter</DisplayName>
        <References>
            <Reference ReferenceType="HasComponent">ns=1;i=5005</Reference>
            <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
            <Reference ReferenceType="Organizes" IsForward="false">i=85</Reference>
            <Reference ReferenceType="HasComponent">ns=1;i=5010</Reference>
        </References>
    </UAObject>
    <UAObject NodeId="ns=1;i=5005" BrowseName="1:VirtuelleKlemmleiste">
        <DisplayName>VirtuelleKlemmleiste</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=58</Reference>
            <Reference ReferenceType="HasComponent">ns=1;i=6012</Reference>
            ...
        </References>
    </UAObject>
    <UAVariable DataType="Double" NodeId="ns=1;i=6012" BrowseName="1:VK_X" AccessLevel="3"
        MinimumSamplingInterval="1000" Historizing="false">
        <DisplayName>VirtuelleKlemmeX</DisplayName>
        <References>
            <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
            <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=5005</Reference>
        </References>
    ...
</UANodeSet>
```

Abbildung 4.12: Ausschnitt aus einem OPC UA Informationsmodell mit integrierter Virtueller Klemmleiste im XML Format

Nicht-standardisierte Eigenschaften, wie der minimale und maximale Variablenwert werden mittels eines Datentyps (*engl.: DataType*[3]) definiert, siehe Abbildung 4.13. Der Datentyp *MinValue* besitzt eine Beschreibung (*engl.: Description*) und ein Feld (*engl.: Field*), welches den minimalen Grenzwert als Zahl (*engl.: Number*) vorsieht. In Analogie zu dieser Datentypdefinition wird auch der maximale Variablenwert einer Virtuellen Klemme in einem OPC UA Informationsmodell spezifiziert.

Nach der vollständigen Erstellung des OPC UA Informationsmodells der Virtuellen Klemmleiste, kann dieses in einen OPC UA Server importiert werden, wodurch eine ausführbare Schnittstelle entsteht, die zur Verknüpfung der Virtuellen Verhaltenslogik mit weiteren Partialmodellen des Digitalen Zwillings verwendet wird. Als ausführbare

[3] Eine Datentypdefinition in OPC UA beschreibt eine Eigenschaft (*engl.: Property*). Eine Property besitzt keine Unterknoten, während mit einer Variablentypdefinition (*engl.: VariableType*) eine OPC UA Struktur spezifiziert werden, die mittels weiterer Unterknoten (Kinder) erweitert werden kann.

```
<DataType SymbolicName="1:MinValue" BaseType="ua:Structure"
    <Description>Datentyp für den unteren Grenzwert einer Variable</Description>
    <Fields>
        <Field Name="MinGrenzwert" DataType="ua:Number">
            <Description>Unterer Grenzwert einer Variable</Description>
        </Field>
    </Fields>
</DataType>
```

Abbildung 4.13: Definition eines Datentyps für die Virtuelle Klemmleiste in OPC UA

Instanz kann die Virtuelle Klemmleiste anschließend konfiguriert und an Datenbanken zur Speicherung und Pufferung angebunden werden. Diese Schritte werden im nachfolgendem Kapitel 4.4 näher beschrieben. Das Produkt des Teilprozesses A133 Generierung der *Virtuellen Klemmleiste* und damit die Ausgabe des Gesamtprozesses *A1* zur Virtualisierung der Verhaltenslogik ist somit eine *Virtuelle Verhaltenslogik mit Virtueller Klemmleiste*.

4.4 Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik

Nachdem der Schwerpunkt des Konzepts mit der Virtualisierung der Verhaltenslogik und Generierung der Virtuellen Klemmleiste im vorigen Kapitel erläutert wurde, erfolgt die Beschreibung der Erweiterung eines Digitalen Zwillings durch die virtualisierte Verhaltenslogik. Im Folgenden wird aufgezeigt, wie die Verhaltenslogik mit weiteren Partialmodellen des Digitalen Zwillings gekoppelt werden kann, um dadurch den tatsächlichen Nutzen einer Virtuellen Klemmleiste praktisch zum Einsatz zu bringen.

Die Zuordnung des zweiten Teils der Integration der Verhaltenslogik in den Digitalen Zwilling in den Gesamtprozess kann der *A0* Ansicht in der Abbildung 4.3 entnommen werden. Die virtualisierte Verhaltenslogik bildet in dem Teilprozess *A2: Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik* eine notwendige Eingangsgröße. Daneben müssen aber auch weiterhin die gestellten Anforderungen und die Partialmodelle in den Teilprozess *A2* mit eingehen.

Als Ressourcen werden Modellierungswerkzeuge benötigt, um erforderlichen Modelle des Digitalen Zwillings zu generieren und Simulationswerkzeuge, um unter Verwendung der generierten Modelle eine ausführbare Instanz zu erreichen. Schließlich werden Kommunikationstechnologien benötigt, um zwischen den eher heterogenen Elementen des Digitalen Zwillings mit integrierter Verhaltenslogik eine bidirektionale Verbindung herzustellen. Das Ergebnis des Gesamtprozesses A0 bildet schließlich der Digitale Zwilling mit integrierter Verhaltenslogik.

Im Folgenden werden die einzelnen Teilschritte des Prozesses *A2* erarbeitet und eine Methode aufgezeigt, die Virtuelle Verhaltenslogik in den Digitalen Zwilling zu integrieren.

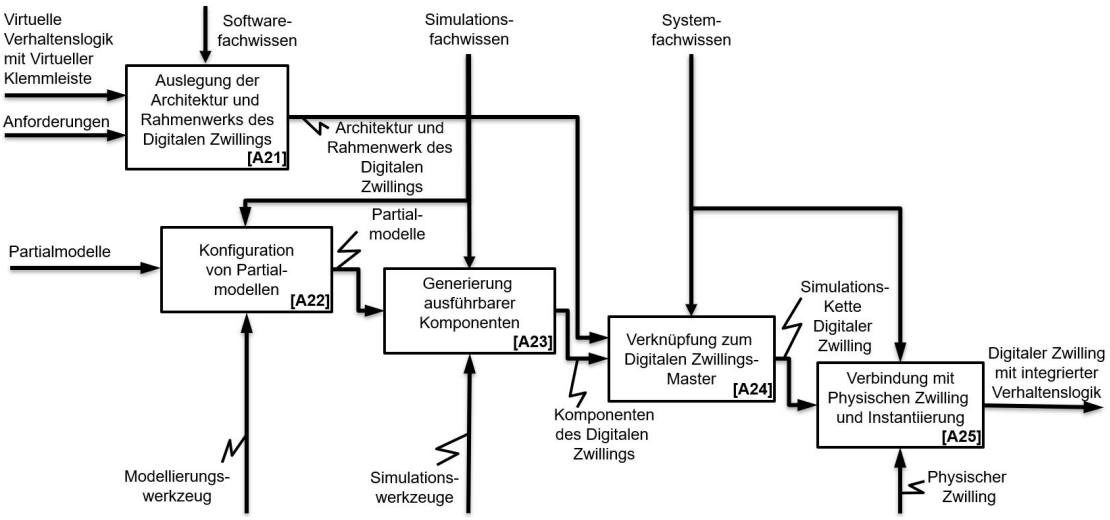

Abbildung 4.14: Teilprozess A2: Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik

4.4.1 Auslegung der Architektur und Rahmenwerks des Digitalen Zwillings

Der erste Schritt *A21* im beschriebenen Prozess zur Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik betrifft die Beschreibung einer für den Anwendungsfall geeigneten Architektur und Rahmenwerks (engl.: *Framework*) des Digitalen Zwillings. Im nachfolgenden wird ein generischer Ansatz entwickelt, welcher in unterschiedlichen, anwendungsfall-spezifischen Architekturen eingesetzt und berücksichtigt werden kann. Im Stand der Technik wurden im Kapitel 2.4 erläutert, dass im Digitalen Zwilling die verteilte und parallele Co-Simulation eine wesentliche Rolle einnimmt und es wurden unterschiedliche Architekturen vorgestellt. Der Aufbau der Architektur ist abhängig von den Domänen und dem modellierten System. Dadurch ist es nicht möglich, eine bereits konkretisierte Architektur zu definieren, welche auf alle Systeme und Anwendungsfälle angewendet werden kann, sondern die speziellen Anforderungen detaillieren eine vorher abstrakte Architektur. Das Rahmenwerk wird aus den Technologien gebildet, die für die Verknüpfung, Orchestrierung, Management, Kommunikation und die Bereitstellung sonstiger Funktionen, welche für die Ausführung des Digitalen Zwillings als Gesamtkonstrukt

benötigt werden, zuständig. Die Architektur und das Rahmenwerk sind eng miteinander verbunden und werden aus diesem Grund im Folgenden gemeinsam betrachtet.

Bei der Auslegung der Architektur eines Digitalen Zwillings kann der Aufbau des physischen Zwillings als Grundlage dienen. Am Beispiel eines mechatronischen Systems, dessen Verhalten im Digitalen Zwilling mit abgebildet werden soll, würden die Elemente *Aktor*, *Sensor*, *Mechanik*, *eingebettete Elektronik*, *eingebettete Software* und *Umgebung* berücksichtigt werden müssen (Abbildung 4.15). Diese Elemente stehen gemäß in Abbildung 2.2 miteinander in Wechselwirkung, welche auch in der Architektur des Digitalen Zwillings eines mechatronischen Systems berücksichtigt werden sollte.

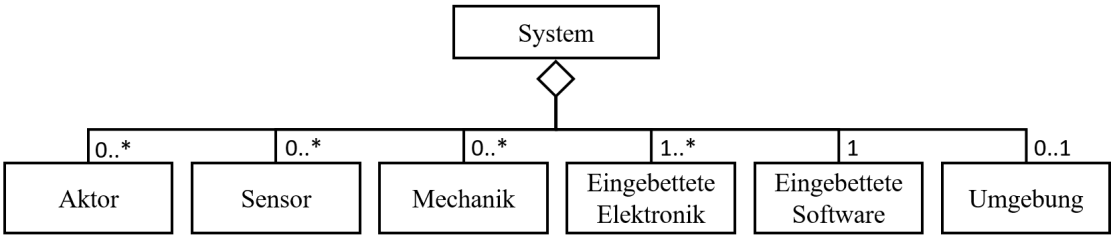

Abbildung 4.15: Übersicht der Modellelemente

Ein weiterer Einflussfaktor neben dem physischen System betrifft die Anforderungen an den Digitalen Zwilling, welche als Eingabe in den Teilprozess *A21* einfließen. Einige generische und beispielhafte Anforderungen, die teilweise aus den vorgestellten Charakteristika des Digitalen Zwillings 2.3.2 abgeleitet werden können, sind *Flexibilität*, *Datenaustauschgeschwindigkeit*, *Kompatibilität in internen- und externe Schnittstellen* und *Sicherheit*.

Die Flexibilität muss in mehrfacher Hinsicht adressiert werden. Einerseits muss der Digitale Zwilling geeignet sein, in der Nutzungsphase beständig weiterentwickelt zu werden und sich zu verändern. Die aus der Softwareentwicklung bekannte Continuous Integration muss ebenso auf den Digitalen Zwilling anwendbar sein, da die Software des Systems nun auch ein Teil des Digitalen Zwillings ist. Andererseits besteht der Digitale Zwilling aus unterschiedlichen Elementen, die miteinander vernetzt sind und interagieren können. Zu diesen Elementen können neben Simulationen auch heterogene Datenbanksysteme und sonstige datenverarbeitende Programme dazugehören. Die Anzahl und Umfang der Elemente können zum Zeitpunkt der Auslegung der Architektur unbekannt sein, sodass hier vorausschauend eine ausreichende Skalierbarkeit eingeplant werden muss, indem eine erweiterbare Architektur gewählt wird. Eine weitreichende Flexibilität würde es erlauben, dass einzelnen Elemente zur Laufzeit hinzugefügt oder wieder entfernt werden können,

wobei diese Anpassungen auch autonom ablaufen können müssen. Dies betrifft auch die Verbindung zum Physischen Zwilling, welcher beispielsweise für Testzwecke entkoppelt werden könnte. In diesem Kontext kann es außerdem sinnvoll sein, mehrere Instanzen des Digitalen Zwillings zu generieren, um zum Beispiel unterschiedliche Konfigurationen testen zu können.

Bei der Auslegung der Architektur muss bereits die geforderte Datenaustauschgeschwindigkeit berücksichtigt werden. Im Kontext der Verhaltensabbildung ist es beispielsweise erstrebenswert, das System möglichst in Echtzeit abzubilden. Dazu ist es notwendig, sehr hohe Datenraten in der Kommunikation mit der virtuellen Elektronik und der eingebetteten Software zu erreichen, da hier hohe Übertragungsfrequenzen erreicht werden können. Falls diese im Digitalen Zwilling nicht unterstützt werden, läuft dieser in der Abbildung des Verhaltens dem physischen System hinterher, wodurch der Nutzen eingeschränkt wird. Im Rahmen der Dissertation wird OPC UA als Kommunikationstechnologie verwendet. OPC UA bietet in Kombination mit *Time-Sensitive Networking*[4] (*OPC UA over TSN*) die Möglichkeit, Daten in der industriellen Produktion herstellerübergreifend in Echtzeit zu übertragen.

In der Planung der Architektur werden weiterhin auch die internen und externen Schnittstellen berücksichtigt. Die internen Schnittstellen betreffen die Kompatibilität der Elemente des Digitalen Zwillings und müssen für den Anwendungsfall geeignet sein. Gute Kompatibilität kann dabei durch Verwendung von standardisierten und offenen Schnittstellen erreicht werden. Diese müssen dabei den heterogenen Datenstrom unterstützen, der sowohl für schnelle Datenübertragung als auch für große Datenvolumen geeignet sein sollte. Im Kontext von Digitalen Zwillingen von Cyber-Physischen Systemen muss weiterhin die externe Konnektivität sichergestellt sein, um in Analogie zum physischen Pendant eine Vernetzungsfähigkeit mit externen Ressourcen sicherzustellen. Im vorgestellten Konzept wird die externe Konnektivität erreicht, indem das Simulationsmodell der virtuellen Elektronik an die Schnittstellen angebunden wird. Dadurch kann in der virtuellen Elektronik beispielsweise eine Ethernet-Schnittstelle implementiert werden, welche in Analogie zur physischen Elektronik die Kommunikation über das Internet ermöglicht. Entscheidend bei der Umsetzung externer Kommunikationskanäle des Digitale Zwilling ist die Unterstützung entsprechender Teilmodelle auf Hardwareebene durch die Virtuelle Elektronik. Bei der Wahl der Elektronik-Simulationsumgebung muss entsprechend berücksichtigt werden, ob alle erforderlichen Kommunikationsschnittstellen des Physischen Zwillings als Teilmodell vorliegen und in der Virtuellen Verhaltenslogik implementiert werden können.

Die aufgezeigten Anforderungen können je nach Anwendungsfall durch weitere Vor-

[4] *TSN* bezeichnet eine Kombination aus mehreren Technologien des Ethernet-Standards IEEE 802.1

gaben ergänzt werden, welche durch die Architektur für den Digitalen Zwilling erfüllt werden müssen. Mögliche Architekturen aus dem Stand der Forschung wurden bereits im Kapitel 2.3.3 näher erläutert. Zu diesen Architekturen gehörten das Referenzmodell Digitaler Zwilling nach Schleich et al. [128], der experimentierbare Digitaler Zwilling nach Schluse et. al. [129], die Open-Source-basierte Architektur nach Damjanovic-Behrend und Behrendt [41], die Referenzarchitektur nach Rovere et al. [40], die Architektur eines Digitalen Zwillings eines Cyber-Physischen Systems nach Zheng und Sivabalan [165] sowie die Referenzarchitektur eines Digitalen Zwillings nach Schroeder et al. [134]. Weiterhin definiert die Verwaltungsschale (vgl.: Kapitel 2.2.2), als eine ebenfalls geeignete Technologie im Kontext des Digitalen Zwillings zusätzliche Rahmenbedingungen, die bei der Auslegung der Architektur eines Digitalen Zwillings berücksichtigt werden können.

Insgesamt wird für die Auslegung der Architektur Fachwissen aus der Softwareentwicklung benötigt, weshalb hier der Softwareingenieur mitwirken muss. Die vorgestellten Anforderungen betreffen ebenso die Auswahl der Technologien eines geeigneten Rahmenwerks. Die Autoren der oben beschriebenen Architekturen des Digitalen Zwillings nennen im Zusammenhang mit der Architektur auch die dabei zum Einsatz kommenden Technologien, weshalb auch hier die Architektur und das Rahmenwerk gemeinsam betrachtet werden. Kutscher et al. schlagen in einer wissenschaftlichen Vorarbeit die Nutzung von web-basierten Technologien für ein neutrales Rahmenwerk des Digitalen Zwillings [85]. Eine weitere bedeutsame Eingangsgröße, die bei der Auslegung der Architektur und Rahmenwerk berücksichtigt werden muss, ist die *Virtuelle Klemmleiste,* die im ersten Prozessteil *A1* entwickelt wurde. Diese ist in der virtuellen Verhaltenslogik bereits enthalten und muss in das Rahmenwerk integriert werden mit dem Ziel, die Schnittstelle zwischen der virtuellen Verhaltenslogik und den davon abhängigen Elementen des Digitalen Zwillings zu bilden. In zukünftigen Betrachtungen ist die Virtuelle Klemmleiste somit auch ein Teil des Rahmenwerks des Digitalen Zwillings. Die Ausgabe des Teilprozesses *A21* bildet demnach die Architektur und Rahmenwerk des Digitalen Zwillings, welche in einem späteren Prozessschritt die Vorlage für die Erweiterung des Digitalen Zwillings bildet.

4.4.2 Konfiguration von Partialmodellen

Im Teilprozess *A22: Konfiguration von Partialmodellen* erfolgt die Anpassung von Partialmodellen, die als Eingabe in den Teilprozess vorgesehen sind, an die Rahmenbedingungen des Digitalen Zwillings. Dabei gilt die Annahme, dass die Partialmodelle aus der Produktentwicklung weiterverwendet werden können. Die Alternative der nachträglichen Generierung durch Retrofitting wurde durch Kutscher et al. beschrieben [84]. Die Par-

tialmodelle, welche im Rahmen der Dissertation hauptsächlich Simulationsmodelle sind, sind Elemente des Digitalen Zwillings, die gemäß der Architektur in den nachfolgenden Prozessschritten zu einem Gesamtsystem verknüpft werden. Im Falle von Simulationsmodellen erfolgt in diesem Prozessschritt die Anpassung der Modelle an das Einsatzszenario des Digitalen Zwillings. Zu diesen Anpassungen gehört z.B. die Ermöglichung einer zeitkritischen und autonomen Ausführung der Simulation, sodass diese in einer kontinuierlichen Simulationskette des Digitalen Zwillings verwendet werden kann. Die Anpassungen hängen vom konkreten Anwendungsfall ab. Dementsprechend muss der Simulationsexperte geeignete Partialmodelle auswählen und diese ggf. modifizieren und konfigurieren. Als Ausgabe werden Partialmodelle benötigt, welche für den Einsatz in einer autonomen, parallelen und verteilten Co-Simulation geeignet sind.

4.4.3 Generierung ausführbarer Simulationen

Aus den Partialmodellen werden im nächsten Prozessschritt *A23* ausführbare und plattformneutrale Programme generiert, die unabhängig von proprietären Werkzeugen, wie beispielsweise Simulationsumgebungen, verwendet werden können. Eine geeignete Technologie, die dabei zum Einsatz kommen kann, ist beispielsweise FMI (vgl. Kapitel 2.4). Mithilfe dieser Technologie können die Simulationswerkzeuge, die als Ressource benötigt werden, standardisierte Simulationseinheiten in Form von FMUs exportieren[5], welche anschließend unabhängig von den Simulationsumgebungen ausgeführt werden können. Detaillierte Ausführungen zum Einsatz von FMUs im Digitalen Zwilling können Kutscher et al. [85] entnommen werden.

Zur Integration der eingebetteten Software in den Digitalen Zwilling muss auch die Elektronik als eine ausführbare Simulation bereitgestellt werden. Auf diesem Bereich kommt der FMI Standard an seine Grenzen, weil FMI in der Domäne der Virtuellen Plattformen aktuell nicht verwendet und nicht unterstützt wird. Demnach kann einen Elektronik-Simulation nicht als eine FMU aus einer Virtuellen Plattform exportiert werden. Eine alternative Möglichkeit ist jedoch die Nutzung von Service-basierten, externen Simulationen, die in einen dezentralen Digitalen Zwilling integriert werden können. Dabei wird eine Simulation in einer proprietären Umgebung ausgeführt und die Ein- und Ausgaben als Service bereitgestellt. Solche dezentralen Elemente des Digitalen Zwillings können demnach in Anwendungsfällen erforderlich sein und müssen in der Architektur und Rahmenwerk berücksichtigt werden.

Die Ausgabe dieses ebenfalls sehr individuellen Prozesses *A23* ist ein im vorgesehenen

[5] Eine Übersicht der Simulationswerkzeuge, die FMI unterstützen, kann www.fmi-standard.org/tools/ entnommen werden.

Rahmenwerk ausführbares Element des Digitalen Zwillings.

4.4.4 Verknüpfung zum Digitalen Zwillings-Master

Nach erfolgter Auslegung der Architektur und des Rahmenwerks des Digitalen Zwillings erfolgt im nächsten Teilprozess *A24* die Verknüpfung der Elemente innerhalb des Rahmenwerks zu einem Digitalen Zwillings-Master, der vorerst keine Verbindung zum Physischen Zwilling aufweist. Damit ist es zunächst ein Master-Modell des Digitalen Zwillings, welches erst im nächsten Schritt instanziiert und mit dem physischen System verknüpft wird.

Die im Kontext der vorliegenden Dissertation bedeutsamen Elemente des Rahmenwerks und des Digitalen Zwillings allgemein sollen im Folgenden hervorgehoben werden. Weitere Herausforderungen, die eine Verknüpfung der Partialmodelle zu einer autonomen, verteilten und parallelen Co-Simulation in einem Rahmenwerk hervorbringt, wie beispielsweise die Diskretisierung von Datenaustausch und Synchronisierung der Simulationseinheiten, können entsprechender Literatur aus dem Kapitel 2.4 entnommen werden.

Die Erstellung eines Co-Simulations-Rahmenwerks ist eine herausfordernde und komplexe Aufgabe, da eine Interoperabilität zwischen den Partialmodellen erreicht werden muss, die einigen, bereits beschriebenen Anforderungen gerecht werden muss. Einige Co-Simulation Technologien, die dabei verwendet werden können sind: die *High Level Architecture* [67], die *Common Object Request Broker Architecture* [113], das *Distributed Co-Simulation Protocol* [103], die Technologie nach der *Open-Service-Gateway Initiative* (OSGi) [117], *Agenten-basierte Systeme* [73, 74] sowie das web-basierte Rahmenwerk für den Digitalen Zwilling nach Kutscher et al. [85]. Insbesondere besitzt der web-basierte Digitale Zwilling eine unterstützende Funktion bei der Verknüpfung der Partialmodelle in Form eines *Simulation Boards*. Plesker et al. führen diese Funktionalität ein, um die einzelnen Simulationselemente des Digitalen Zwillings zu einer durchgehenden Simulationskette verknüpfen zu können. Eine weitere Technologie, die für eine standardisierte und insbesondere semantische Verknüpfung der Elemente des Digitalen Zwillings genutzt werden kann, ist OPC UA. Neben der Beschreibung des Einsatzes von OPC UA in der Virtuellen Klemmleiste in Kapitel 4.3.3 kann Kutscher et al. entnommen werden, wie ein von den Companion Specifications abgeleitetes OPC UA Informationsmodell genutzt werden kann, um in Analogie zur physischen System die Elemente des Digitalen Zwillings miteinander zu verbinden [86]. Dadurch können die Co-Simulationen mittels gleicher Schnittstellen gekoppelt werden, wie es bei den physischen Komponenten der Fall ist. Die semantische Beschreibung dieser Schnittstelle bringt dabei den Vorteil gegenüber anderen Technologien mit sich, dass die Kopplung auf standardisierten Komponentenbeschreibungen (Companion Specifications) basiert und somit autonom durchgeführt werden kann.

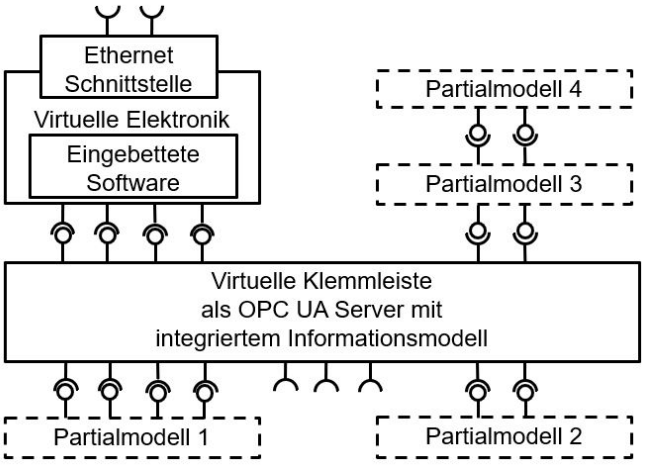

Abbildung 4.16: Verknüpfung der Modelle des Digitalen Zwillings-Masters mittels der in OPC UA umgesetzten Virtuellen Klemmleiste

Zur Verdeutlichung sind in der Abbildung 4.16 die Verknüpfungen der Modelle des Digitalen Zwillings-Masters dargestellt. Die eingebettete Software wird in der virtuellen Elektronik ausgeführt, welche gemäß dem Kapitel 4.3.2 mittels einer Simulation virtualisiert wurde. Die Elektroniksimulation beinhaltet in dem Beispiel ein Teilmodell einer Ethernet Schnittstelle, die zur Kommunikation mit externen Systemen genutzt wird. Die virtuelle Elektronik ist außerdem mit der Virtuellen Klemmleiste verknüpft. Die Virtuelle Klemmleiste wurde als ein OPC UA Informationsmodell modelliert in ein OPC UA Server integriert, wodurch es eine ausführbare Schnittstelle bildet. Mittels der Virtuellen Klemmleiste steht die Verhaltenslogik bereit, um mit weiteren Partialmodellen des Digitalen Zwillings verknüpft zu werden. Die Virtuelle Klemmleiste kann auch verwendet werden, um unabhängig von der Verhaltenslogik Partialmodelle des Digitalen Zwillings zu verknüpfen - was jedoch nicht der Fokus der vorliegenden Dissertation ist.

4.4.5 Verbindung mit Physischen Zwilling und Instanziierung

Im Teilprozess *A25: Verbindung mit Physischen Zwilling und Instanziierung* wird schließlich die bidirektionale Verbindung zwischen den beiden Zwillingen hergestellt. Damit kommt das wesentliche Merkmal des Digitalen Zwillings, welches in der ständigen Wechselwirkung mit dem physischen System besteht, erstmals zum Tragen. In diesem Schritt wird aus dem Digitalen Zwillings-Master, welcher vorher noch ein neutrales Modell ist, ein

oder mehrere Digitale Zwillinge instanziiert, welche durch die Verknüpfung mit dem Physischen Zwilling dessen besondere Eigenschaften und Systemzustände annehmen. Die Möglichkeit, mehrere Instanzen des Digitalen Zwillings zu bilden wird an dieser Stelle hervorgehoben, da im Rahmen von Anwendungsfällen, wie beispielsweise das Testen unterschiedlicher Konfigurationen, diese Fähigkeit bedeutsam ist.

Die bidirektionale Verbindung zwischen dem Physischen und Digitalen Zwilling wird hergestellt, indem gezielt Daten zwischen den Systemen ausgetauscht werden. Die virtuelle Verhaltenslogik erhält im parallelen Betrieb nun Eingabedaten vom physischen System. Zu diesen gehören die Steuerungsdaten, die gleichzeitig auch an die Verhaltenslogik des Physischen Zwillings gesendet werden. Je nach Konfiguration des Digitalen Zwillings fließen außerdem entweder simulierte oder reale Sensordaten in die Partialmodelle und die Virtuelle Verhaltenslogik. Damit werden die in der eingebetteten Software implementierten Regelungen mit Eingabedaten versehen und generieren entsprechende Ausgabedaten, die an anschließende Partialmodelle des Digitalen Zwillings weitergeleitet und dort verarbeitet werden. Umgekehrt werden die Auswertungen und Erkenntnisse aus dem Digitalen Zwilling auf das physische System angewendet. Zu diesen Erkenntnissen gehören beispielsweise errechnete bevorstehende Verhaltensweisen des Systems, die aus dem aktuellen Zustand mit prädiktiven Modellen vorhergesagt werden können. Damit ist die bidirektionale Verbindung zwischen den beiden Zwillingen hergestellt.

Die Ausgabe des finalen Teilprozesses *A25* ist ein Digitaler Zwilling mit integrierter Verhaltenslogik. Somit ist der Gesamtprozess abgeschlossen, in welchem zunächst die Verhaltenslogik virtualisiert, eine *Virtuellen Klemmleiste* generiert und anschließend in das Gesamtkonstrukt des Digitalen Zwillings integriert wurde.

4.5 Fazit

Kapitel 4 beinhaltet die Konzeption einer Vorgehensweise zur Integration des Verhaltens in einen Digitalen Zwilling. Das Konzept ist untergliedert in die zwei Themenbereiche *Virtualisierung der Verhaltenslogik* und *Erweiterung des Digitalen Zwillings durch virtualisierte Verhaltenslogik*.

Im ersten Teil des entwickelten Prozesses wird beschrieben, wie die eingebettete Verhaltenslogik des physischen Systems virtuell abgebildet werden kann, dass diese für die Nutzung im Digitalen Zwilling grundsätzlich geeignet ist. Dazu zählt insbesondere die Analyse der Verhaltenslogik, die Virtualisierung der eingebetteten Elektronik und die Generierung einer informationsmodell-basierten Schnittstelle mittels der OPC UA Technologie, welche als Virtuelle Klemmleiste bezeichnet wird. Mithilfe der *Virtuellen Klemmleiste* kann im zweiten Teil des Konzepts der Physische Zwilling und der Digitale Zwilling mit-

einander verknüpft werden. Dazu wird zunächst eine generische Architektur aufgezeigt und anschließend die Prozessschritte bei der Erweiterung des Digitalen Zwillings durch die virtuelle Verhaltenslogik beschrieben.

5 Prototypische Implementierung

Im nachfolgenden Kapitel wird die prototypische Implementierung des entwickelten Konzepts einer Virtuellen Klemmleiste zur Integration der Verhaltenslogik in den Digitalen Zwilling vorgestellt. Die Struktur des Kapitels orientiert sich am Aufbau des Konzeptteils und behandelt zunächst die Virtualisierung der Verhaltenslogik und anschließend die Erweiterung eines Digitalen Zwillings durch die virtualisierte Verhaltenslogik. Als ein repräsentativer Demonstrator und Physischer Zwilling dient ein CNC Laserplotter, dessen Hardware und Software die Grundlage der Erweiterung des Digitalen Zwillings bildet.

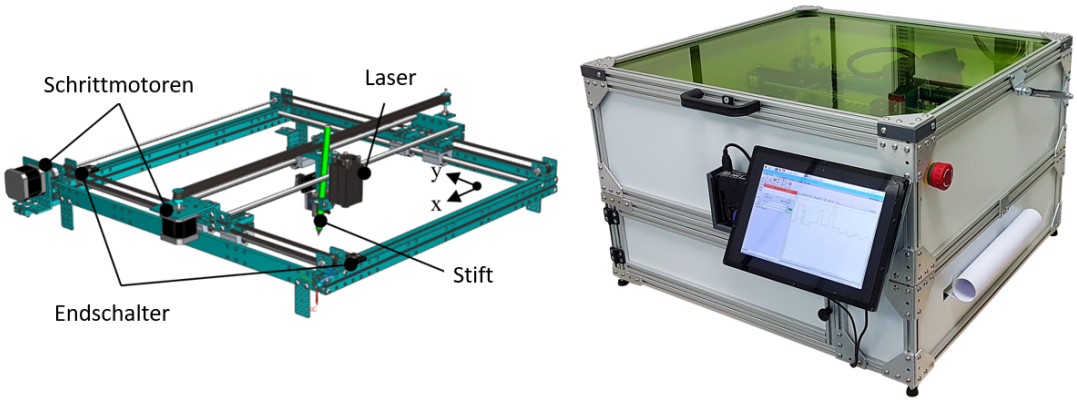

Abbildung 5.1: CNC Laserplotter als Demonstrator

Der CNC Laserplotter ist in der Abbildung 5.1 dargestellt. Die Mechanik des CNC Laserplotters besteht aus Aluminiumprofilen, die durch Führungen, einem Riementrieb und Schrittmotoren ergänzt werden. An den Anschlägen der einzelnen Achsen befinden sich Endschalter. Die Endschalter stellen die Sensoren des Systems dar, welche die Randposition der einzelnen Achsen erfassen und an die Steuerung signalisieren. Der CNC Laserplotter ist weiterhin mit einem Laser ausgestattet, der zum Schneiden und Gravieren auf geeigneten Materialien wie Papier, Holz, Leder oder Ähnlichem verwendet werden kann. Die Mechanik des Laserplotters ist in einem Gehäuse integriert, welches

aus Aluminiumprofilen und Sandwichplatten aufgebaut ist. Weiterhin ist in dem Gehäuse ein Papiertransportmechanismus (Papier-Conveyor) integriert, welcher Papier von einer Vorratsrolle über die Arbeitsfläche befördert und seitlich aus dem Gehäuse führt.

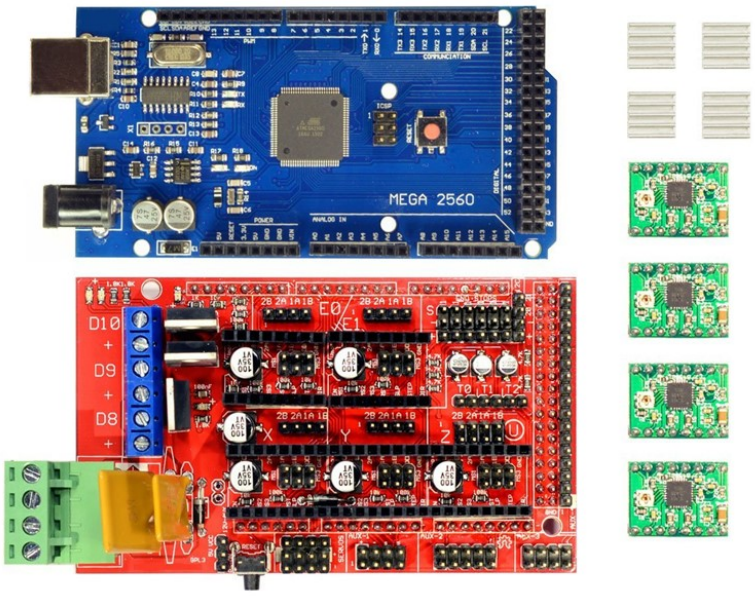

Abbildung 5.2: Elektronische Komponenten des CNC Laserplotter. MEGA2560 Board (oben links), das RAMPS 1.4 (unten links) Board und die Pololu DRV8838 Motortreiber mit Kühlkörpern (rechts) im demontierten Zustand.

Die Verhaltenslogik des Systems wird durch die Elektronik und die eingebettete CNC Steuerungssoftware des Laserplotters realisiert (siehe Abbildung 5.2). Die Steuerungssoftware ist in einem *Microchip ATMEGA 2560* Mikrocontroller eingebettet. Der Mikrocontroller bildet wiederum die zentrale Recheneinheit eines *Arduino MEGA2560 Boards*. Der *Arduino MEGA2560* ist mit der Leistungselektronik unter der Bezeichnung *RAMPS 1.4* (*RepRap Arduino MEGA Pololu Shield*) verbunden, welche zudem die *Pololu* Motortreiber *DRV8838* integriert. Als CNC Steuerungssoftware wird der quell-offene G-Code Interpreter *Grbl-Mega* eingesetzt [37]. *Grbl-Mega* ermöglicht die Steuerung von CNC Systemen die einen MEGA2560 Board verwenden. Die umgangssprachlich als *G-Code* bezeichnete CNC Programmiersprache (auch *RS-274*) dient dazu, Maschinenanweisungen in einer überwiegend standardisierten Form zu beschreiben. Der CNC Laserplotter enthält weiterhin eine Benutzungsoberfläche mit der Bezeichnung *bCNC* [160], die auf einem seriell verbunde-

nen Computer (*Raspberry Pi 3B+*) ausgeführt wird. Diese Mensch-Maschine Schnittstelle erlaubt sowohl die direkte Steuerung des CNC Laserplotters, als auch die Vorgabe von kompletten Bearbeitungsaufgaben in Form von G-Code Daten oder Vektorgraphiken. Die Vektorgraphiken werden durch bCNC zu G-Code Anweisungen übersetzt und über eine serielle Verbindung Zeile für Zeile an die eingebettete Elektronik übermittelt.

Auf Basis dieses Demonstrators wird im Folgenden eine Virtualisierung der Verhaltenslogik des Systems durchgeführt und anschließend der Digitaler Zwilling durch die virtualisierte Verhaltenslogik erweitert. Die prototypische Implementierung dient der Überprüfung des vorgestellten Konzepts hinsichtlich der Tragfähigkeit und bildet gemeinsam mit den im Kapitel 3 vorgestellten Anwendungsfällen die Grundlage für die anschließende Validierung.

5.1 Virtualisierung der Verhaltenslogik

Der im Konzeptteil vorgestellte Prozess beginnt mit der Virtualisierung der Verhaltenslogik des Physischen Zwillings. Dazu wird auf Basis der initialen Analyse der Verhaltenslogik zunächst die Elektronik des Systems durch geeignete Softwarewerkzeuge virtualisiert. Die virtuelle Elektronik wird anschließend genutzt, um die eingebettete Software des Systems auf der Zielrechnerarchitektur des Digitalen Zwillings auszuführen und diese mittels einer *Virtuellen Klemmleiste* mit den Elementen des Digitalen Zwillings zu koppeln. In den nachfolgenden Unterkapiteln werden die Teilprozesse *A1: Virtualisierung der Verhaltenslogik* auf den CNC Laserplotter angewendet.

5.1.1 Analyse der Verhaltenslogik

Die Verhaltenslogik des CNC Laserplotters manifestiert sich in der eingebetteten Elektronik und der eingebetteten Software. Diese bestimmen die Wahl der Softwarewerkzeuge, welche für die Virtualisierung verwendet werden können. Abhängig von der Architektur des Mikroprozessors und restlicher elektronischer Komponenten muss eine kompatible Modellierungs- und Simulationsumgebung gewählt werden. Zur Zeit der Erstellung der Dissertation existiert keine Umgebung, die alle Arten von Mikroelektronik unterstützt, sondern die unterschiedlichen Werkzeuge können für einzelne Mikroprozessorfamilien bzw. wenige Mikroprozessorarten herangezogen werden. Wie oben beschrieben, handelt es sich bei dem Mikroprozessor des CNC Laserplotters um einen *Microchip ATMEGA 2560*, welcher durch die *RAMPS 1.4* Leistungselektronik in Kombination mit *Pololu DRV8838* Motortreibern eingesetzt wird. Diese Komponenten stellen individuelle Anforderungen hinsichtlich einer geeigneten Modellierungs- und Simulationsumgebung.

Weitere Rahmenbedingungen resultieren aus der eingebetteten Software. Die verwendete Modellierungs- und Simulationsumgebung muss imstande sein, diese Software auf der virtualisierten Elektronik auszuführen. Dabei können die Modellierungs- und Simulationsfunktionen in einer gemeinsamen Umgebung integriert sein. Die Steuerungssoftware des CNC Laserplotters *Grbl-Mega* ist in der Programmiersprache *C* geschrieben. Mögliche Programmiersprachen werden durch die Mikroprozessorarchitektur bereits vorgegeben und müssen in der Modellierungs- und Simulationsumgebung einsetzbar sein. Doch nicht nur die Programmiersprache muss berücksichtigt werden, sondern auch die implementierten Funktionen und nicht-funktionale Rahmenbedingungen der Software. Ein wichtiger Faktor ist der *Takt* des Mikrocontrollers. Gemeinsam mit der Komplexität der Software bestimmen diese unter anderem die erreichbare Verarbeitungsgeschwindigkeit durch das Simulationswerkzeug. Falls die Ausführungsgeschwindigkeit des Simulators langsamer ist, als die Verarbeitung im Physischen Zwilling, können Echtzeitanforderungen nicht erfüllt werden und der Digitale Zwilling kann insbesondere nicht parallel zum physischen System betrieben werden.

Weitere individuelle Anforderungen werden an die Kommunikationsschnittstellen des Simulationswerkzeugs gestellt, da hier hohe Übertragungsgeschwindigkeiten bedeutsam werden können. Im Falle des Demonstrators werden in der Steuerungssoftware unter anderem die Steuerbefehle für die Schrittmotoren generiert. Die maximale Winkelgeschwindigkeit und die Schrittzahl pro Winkel erfordern eine schnelle Taktung der Datenübertragung über die Schnittstellen des Simulationswerkzeugs. Eine nicht ausreichende Übertragungsgeschwindigkeit würde die Simulation insgesamt verlangsamen.

Die vorgestellten Anforderungen wurden im Rahmen der Analyse der Verhaltenslogik ermittelt und bestimmen die Wahl der Werkzeuge in der nachfolgenden Virtualisierung der Elektronik. Die möglichen elektronischen Komponenten eines mechatronischen Systems erfordern eine stets individuelle und anwendungsfallabhängige Analyse durch einen Experten mit geeignetem Softwarefachwissen.

Das Ergebnis des Teilprozesses ist die bereitgestellte Verhaltenslogik bestehend aus der eingebetteten Software und die Spezifikation der Hardware- und Softwareeigenschaften.

5.1.2 Virtualisierung der Elektronik

Im nächsten Teilprozess wird die eingebettete Elektronik des CNC Laserplotters mittels geeigneter Modellierungswerkzeuge virtualisiert. Im Rahmen der Implementierung wurden unterschiedliche Werkzeuge hinsichtlich der funktionalen Eignung untersucht. Zu den analysierten Werkzeugen gehörten unter anderem *Microchip Studio* (ehemals *Atmel Studio*) [100] der Firma *Microchip Technology Inc.*, *SimulAVR* [149] und *Proteus 8 Demonstration* der Firma *Labcenter Electronics* [87]. Nach einer Evaluation der Anforderungen wurde

Proteus verwendet, um die eingebettete Elektronik zu virtualisieren. Ausschlaggebende Kriterien bei der Evaluierung waren die Verfügbarkeit der benötigten elektronischen Modelle, die Möglichkeit einer Echtzeitsimulation und die Verfügbarkeit geeigneter Schnittstellen zu externen Werkzeugen.

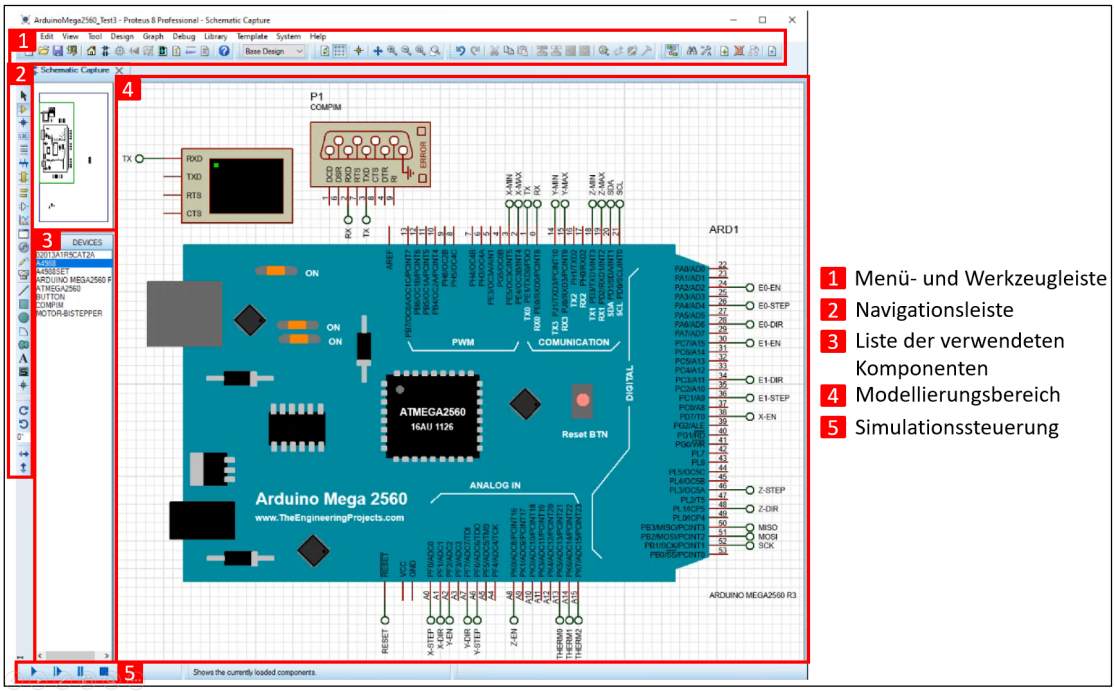

Abbildung 5.3: Modellierung- und Simulationswerkzeug für elektronische Komponenten *Proteus* von *Labcenter Electronics*

Abbildung 5.3 zeigt einen exemplarischen Ausschnitt aus der Modellierungs- und Simulationsumgebung *Proteus* und hebt fünf Bereiche der graphischen Benutzungsoberfläche hervor. Zur Bedienung allgemeiner Funktionen des Programms wird die *(1) Menü- und Werkzeugleiste* verwendet. Die Navigation zwischen einzelnen Modellierungsfunktionen der Benutzungsoberfläche erfolgt mittels der *(2) Navigationsleiste*. Aus der *(3) Liste der verwendeten Komponenten* können Modelle elektronischer Komponenten ausgewählt und im *(4) Modellierungsbereich* platziert werden. Die Liste kann durch Import der Komponentenmodelle sowohl aus einer internen Datenbank, als auch aus externen Datenbanken von Drittanbietern erweitert werden. Dadurch kann auf eine große Anzahl von Modellen elektronischer Komponenten zurückgegriffen werden. Die Auswahl reicht von Modellen

einfacher Komponenten wie beispielsweise von Widerständen und Leuchtdioden, bis zu komplexen Modellen von Mikrocontrollern oder vollständigen Mikrocontroller Boards. In Abbildung 5.3 ist beispielsweise das *Arduino MEGA 2560 Board* als ein einzelnes Modell visualisiert, welches jedoch intern aus wiederum unterschiedlichen Teilmodellen besteht. Zusätzlich zu dem Mikrocontroller Board Modell sind ein Modell einer seriellen Schnittstelle *COMPIM (COMunication port Physical Interface Model)* und ein virtuelles Terminal (engl. *Virtual Terminal*) im Modellierungsbereich abgebildet. Die Modelle der elektronischen Komponenten werden in Analogie zu physischen Komponenten über Verbindungspunkte miteinander gekoppelt. Zur Verbesserung der Übersichtlichkeit werden die Leitungen graphisch nicht dargestellt, sondern über gleiche Benennung der Endpunkte realisiert. Die in der Abbildung 5.3 dargestellten digitalen und analogen Ein- und Ausgänge des virtuellen Mikrocontroller Boards entsprechen den digitalen und analogen Ein- und Ausgängen (*GPIOs - General Purpose Input Output*) des physischen Boards [1]. Über diese Verbindungen werden die Teilmodelle miteinander verknüpft und anschließend durch einen Knopfdruck auf *Run Simulation* im Bereich der *(5) Simulationssteuerung* ausgeführt.

Die Simulation der virtuellen Elektronik ohne eingebettete Software wird im vorgestellten Anwendungsfall einen statischen Zustand annehmen, da die Software für die Schaltvorgänge in der Elektronik zuständig ist. Im nächsten Schritt erfolgt deshalb die Integration der eingebetteten Software und die Konfiguration der Elektronikmodelle. In der Abbildung 5.4 sind zwei Oberflächen dargestellt, mittels derer die Konfiguration der Komponentenmodelle stattfindet. Die *(1) Integration der eingebetteten Software* erfolgt über *PROGRAM FILE* des Konfigurationsfensters des Mikrocontrollers. Damit wird vorgegeben, welche eingebettete Software zur Simulationszeit der virtuellen Elektronik ausgeführt werden soll. Diese Konfiguration entspricht in Analogie der Installation der eingebetteten Software auf dem Physischen Zwilling. Hierfür wird die Quellsoftware vorher kompiliert und damit in die Maschinensprache übersetzt. Weiterhin erfolgt die *(2) Konfiguration des simulierten Arbeitstakts* des virtuellen Mikrocontrollers, welche in der vorgestellten Implementierung dem Arbeitstakt des physischen Mikrocontrollers entspricht.

Weiterhin muss die serielle Kommunikationsschnittstelle des Mikrocontroller Boards konfiguriert werden. Hierfür wird die serielle Schnittstelle des virtuellen Mikrocontrollers, die als *TX (Transmission Exchange)* und *RX (Recieving Exchange)* bezeichnet wird, mit dem Modell einer seriellen Schnittstelle *COMPIM* des Simulationsrechners verbunden (siehe Abbildung 5.3). In der *(3) Konfiguration des Kommunikationsports (Abbildung 5.4)* kann anschließend definiert werden, an welchen physischen COM Port des Simulationsrechners die virtuelle Schnittstelle angebunden wird. In der Abbildung 5.4 findet die Kommuni-

[1] GPIOs sind Anschlüsse, die frei belegt werden können und sind Teil der allgemeineren Bezeichnung *PIN*. Ein PIN ist allgemein ein Anschluss einer integrierten Schaltung

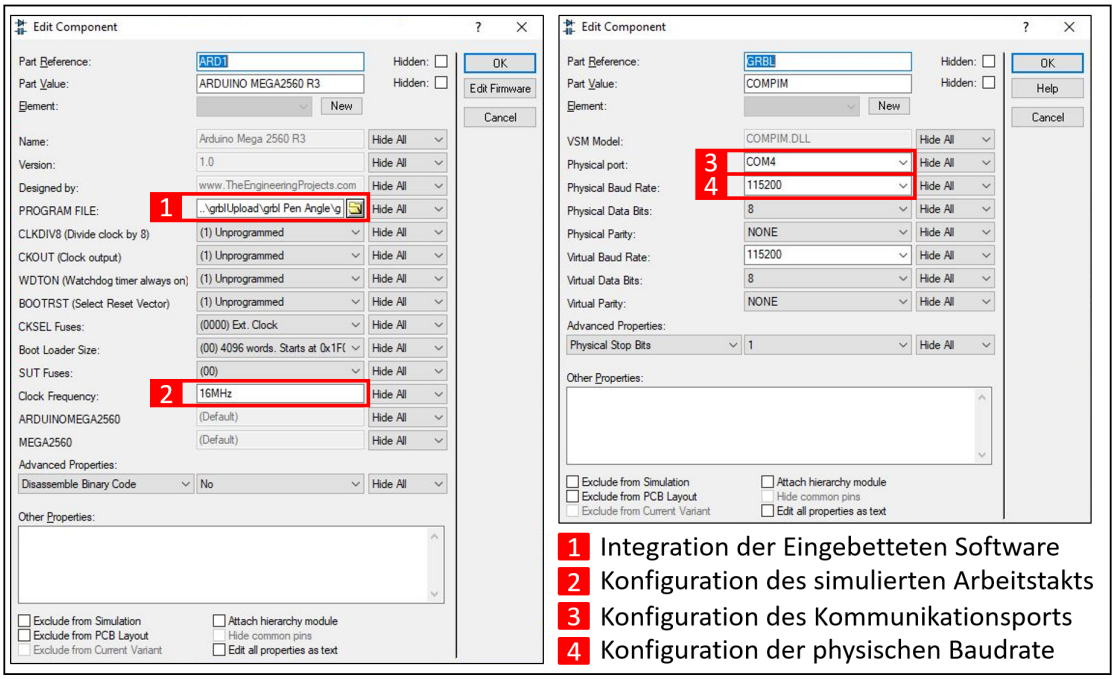

Abbildung 5.4: Konfiguration des Mikrocontrollers *ATMEGA2560* (links) und der Kommunikationsschnittstelle *COMPIM* in *Proteus* von *Labcenter Electronics*

kation über den Port *COM4* statt. Zusätzlich erfolgt die *(4) Konfiguration der physischen Baudrate* der Kommunikation. Damit wird sichergestellt, dass beide Kommunikationspartner die gleiche Frequenz bei Übermittlung und Empfang der Nachrichten verwenden. Das Modellierungswerkzeug bietet weitere Möglichkeiten zur Konfiguration des Modells der virtuellen Elektronik, die der Dokumentation entnommen werden können [87].

Im Anschluss an die Modellierung der virtuellen Elektronik muss eine ausführbare Simulation generiert werden. Je nach verwendeten Modellierungs- und Simulationswerkzeugen ist dieser Schritt manuell auszuführen oder die Modellierung und Simulation findet in einer integrierten Entwicklungsumgebung statt. Das verwendete Softwarewerkzeug *Proteus* kombiniert beide Funktionen und die hinterlegten Simulationsmodelle der einzelnen Komponentenmodelle werden automatisiert ausgeführt.

5.1.3 Virtuelle Klemmleiste

Die Virtuelle Klemmleiste bildet die Verknüpfung zwischen dem Physischen- und dem Digitalen Zwilling, siehe Abbildung 5.5. Der physische Teil des Zwillingssystems enthält die Verhaltenslogik in Form der physischen Elektronik mit eingebetteter Software. An dieser Elektronik sind unterschiedliche *Komponenten 1..n*, wie beispielsweise Sensoren oder Aktoren, angebunden. Über die Virtuelle Klemmleiste erfolgt die bidirektionale Verbindung zu dem Digitalen Zwilling. Die Virtuelle Klemmleiste basiert dabei gemäß dem Konzept auf einem Informationsmodell, welche die Verknüpfung vereinheitlicht. Die Virtuelle Klemmleiste wird als Teil des Digitalen Zwillings betrachtet und die Systemgrenze entsprechend definiert. Innerhalb des Digitalen Zwillings wir die virtuelle Verhaltenslogik ebenfalls an die Virtuelle Klemmleiste angebunden. Die einzelnen Partialmodelle des Digitalen Zwillings, wie beispielsweise Simulationen, können sowohl über die virtuelle Elektronik, als auch direkt mit der Virtuellen Klemmleiste verknüpft werden. In den nachfolgenden Kapiteln erfolgt die Implementierung dieser Virtuellen Klemmleiste.

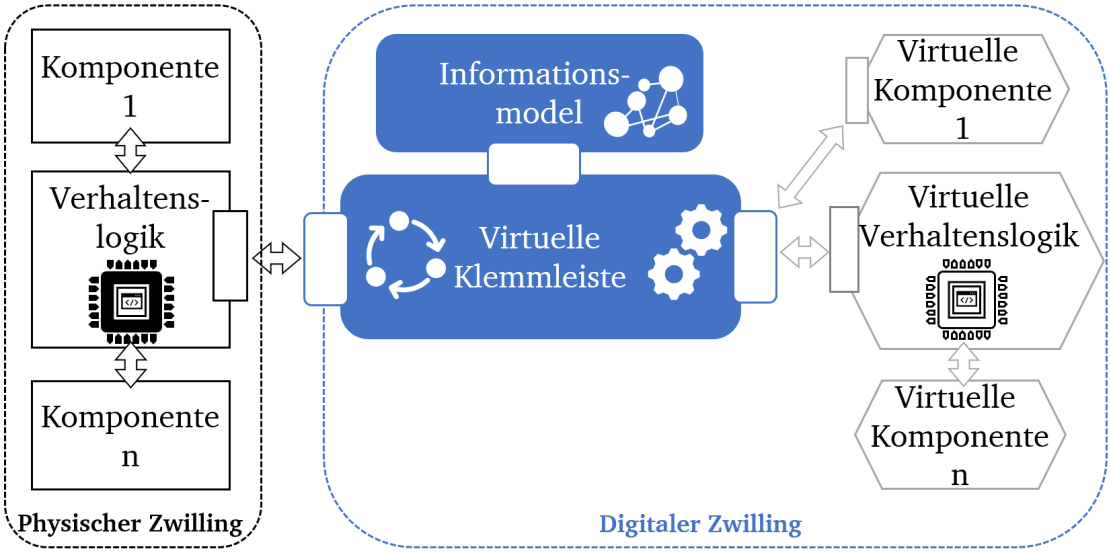

Abbildung 5.5: Architektur des prototypischen Demonstrators

Analyse des Physischen und des Digitalen Zwillings

Die Virtuelle Klemmleiste muss zunächst ausgelegt und konkretisiert werden. Dies erfolgt einerseits durch eine Analyse des Physischen Zwillings und andererseits durch die

Analyse der Partialmodelle des Digitalen Zwillings. Die Analyse des Physischen Zwillings wird gemäß konzeptioneller Beschreibung im Teilprozess A13 aus dem Kapitel 4.3.3 durchgeführt. Im vorliegenden Fall wurde betrachtet, aus welchen Komponenten der CNC Laserplotter zusammengesetzt ist. Zu diesen Komponenten gehören Aktoren, Sensoren, Mechanik, Elektronik, eingebettete Software sowie die Umgebung (vgl. Abbildung 4.15). Die Aktoren und Sensoren sind im vorliegenden Fall mit der eingebetteten Elektronik über definierte GPIOs verbunden. In der eingebetteten Software sind die Verknüpfungspunkte bekannt und die Datenströme werden entsprechend verarbeitet. Für die sinnvolle Anordnung der Modelle in der virtuellen Verhaltenslogik und die Definition der *Virtuellen Klemmleiste* müssen die notwendigen Informationen aus unterschiedlichen Quellen, wie CAD (*Computer Aided Design*)-Modellen, technischen Datenblättern, Schaltplänen und aus der eingebetteten Software und dessen Dokumentation entnommen werden.

Auf der virtuellen Seite des Zwillingssystem befinden sich Partialmodelle des Digitalen Zwillings. Im Folgenden wird vorausgesetzt, dass die Partialmodelle bereits vorliegen. Die Partialmodelle des Digitalen Zwillings können unterschiedliche Teilfunktionen abdecken. Die Bereiche der eingebetteten Elektronik und der eingebetteten Software wurden wie oben beschrieben im Sinne der virtuellen Verhaltenslogik bereits virtualisiert. Nun müssen weitere Teilmodelle an die virtuelle Verhaltenslogik angebunden werden. So wie ein Aktor in einem physischen System die Steuerbefehle von der Verhaltenslogik erhält, muss auch der virtuelle Aktor an die virtuelle Verhaltenslogik angebunden werden. Bei der Implementierung der *Virtuellen Klemmleiste* können bezüglich der Berücksichtigung bereits existierender Partialmodelle unterschiedliche Strategien verfolgt werden. Einerseits kann die Schnittstelle seitens der *Virtuellen Klemmleiste* fest vorgegeben werden und die Schnittstellen der Partialmodell müssen daran ausgerichtet werden oder die Schnittstellen der Partialmodelle sind fest vorgegeben und die Virtuelle Klemmleiste muss entsprechend definiert werden. Im Hinblick auf eine mögliche Standardisierung der Virtuellen Klemmleiste als eine generische Schnittstelle zwischen einem Digitalen- und Physischen Zwilling wurde im Rahmen der folgenden Implementierung eine feste Vorgabe der Virtuellen Klemmleiste und die Anpassung der Schnittstellen der Partialmodelle gewählt.

Nach erfolgter Analyse des physischen Systems und der Partialmodelle kann die Virtuelle Klemmleiste nun weiterentwickelt werden. Die nachfolgenden Erläuterungen zur Implementierung auf Basis des CNC Laserplotters setzten voraus, dass das System im Detail bekannt ist.

OPC UA Informationsmodell der Virtuellen Klemmleiste

Den Kern der Implementierung der Virtuellen Klemmleiste bildet ein OPC UA Informationsmodell. Dieses definiert, welche Verbindungspunkte zwischen dem Digitalen- und

Physischen Zwilling existieren und wie diese miteinander in Beziehung stehen. Für die prototypische Implementierung wurde ein OPC UA Informationsmodell generiert, dessen Ausschnitt in der Abbildung 5.6 dargestellt ist. Als Vorlage dient eine OPC UA Companion Specification für Werkzeugmaschinen der UMATI (*Universal Machine Technology Interface*) Initiative [153]. Die UMATI Companion Specification definiert eine Objekttypen und Komponenten, die im Kontext von Werkzeugmaschinen benötigt werden. Die Vorlage wurde für den Anwendungsfall im Rahmen der Implementierung erweitert, um die Kommunikation zwischen einem Physischen- und einem Digitalen Zwilling zu ermöglichen. Die Erweiterung ist notwendig, da das UMATI Informationsmodell grundsätzlich für eine Maschine-zu-Maschine Kommunikation zwischen physischen Systemen entwickelt wurde und im Folgenden auf den Digitalen Zwilling ausgeweitet wird. Ein Ausschnitt aus dem weiterentwickelten Informationsmodell in der Abbildung 5.6 dargestellt.

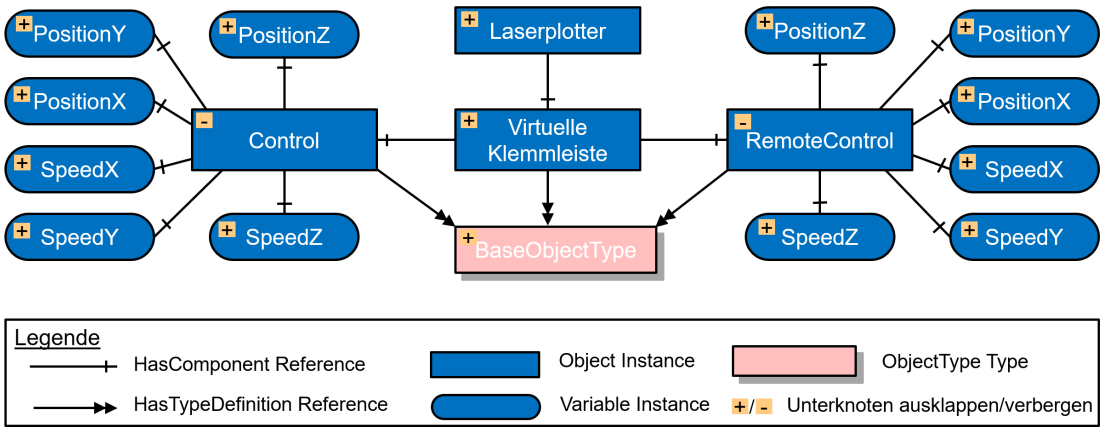

Abbildung 5.6: Ausschnitt des OPC UA Informationsmodells des CNC Laserplotters.

Neben der Definition der einzelnen Achsenpositionen und Geschwindigkeiten im *Control* Bereich des Knotens *Laserplotter* wurde eine zweite Struktur *RemoteControl* definiert, welche den Zustand des Digitalen Zwillings repräsentiert. Sowohl *Control* als auch *RemoteControl* sind Objekte der Virtuellen Klemmleiste. Zur Veranschaulichung ist *BaseObjectType* als die Typdefinition entsprechender Objekte dargestellt. Alle anderen Knoten und Referenzen sind verborgen und können in dem verwendeten Modellierungswerkzeug mittels des Pluszeichens eingeblendet bzw. mit dem Minuszeichen in der oberen linken Ecke des Objekts verborgen werden. Diese Datenstruktur ermöglicht es, dass sowohl der aktuelle Zustand des physischen Systems, als auch der simulierte Zustand des Digitalen Zwillings im OPC UA Informationsmodell gleichzeitig gespeichert und genutzt werden können.

Die Modellierung des OPC UA Informationsmodells erfolgte mithilfe des Softwarewerkzeugs *UA*[2] *Modeller* des Softwareherstellers *Unified Automation GmbH* [152]. Der *UA Modeller* ermöglicht es, OPC UA Informationsmodelle zu importieren, zu verändern und zu exportieren. Demnach konnte die UMATI Companion Specification importiert und erweitert werden. In Abbildung 5.7 ist die graphische Benutzungsoberfläche des *UA Modellers* dargestellt.

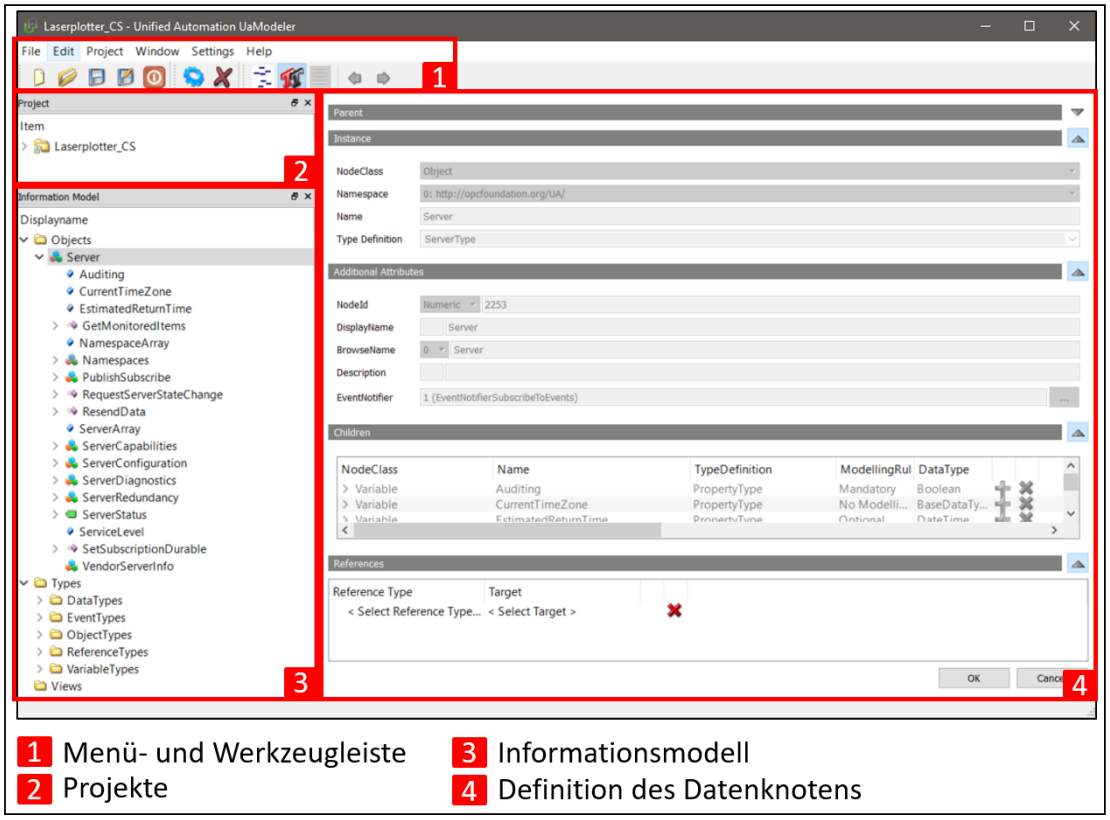

Abbildung 5.7: *UA Modeller* des Herstellers *Unified Automation GmbH*

Neben einer *(1) Menü- und Werkzeugleiste* können mehrere Informationsmodelle in *(2) Projekte* angelegt werden. Ein *(3) Informationsmodell* wird in einer hierarchischen Ansicht dargestellt und kann navigiert werden. Die Daten werden in einem Bereich für

[2]Das UA steht in dem Kontext der Softwarewerkzeuge *UA Modeller* und *UA Expert* für den Namen des Herstellers *Unified Automation GmbH*

(4) Definition des Datenknotens angezeigt und können neu erstellt und modifiziert werden. Mittels der UMATI Vorlage und des *UA Modellers* wurde für den CNC Laserplotter ein Informationsmodell erstellt und im XML Format exportiert. In dem Informationsmodell werden sowohl statische Daten, wie beispielsweise der Hersteller oder Standort, auch Datenpunkte für dynamische Informationen, wie z.B. die Positionen und Geschwindigkeiten der einzelnen Achsen des Systems, definiert.

OPC UA Server der Virtuellen Klemmleiste

Das generierte OPC UA Informationsmodell muss in einen OPC UA Server integriert werden, um als eine ausführbare Instanz in der Virtuelle Klemmleiste als Schnittstelle zu dienen. Hierfür wurde eine OPC UA Server-Client Softwarebibliothek eingesetzt. Es existieren sowohl kommerzielle, als auch quell-offene Implementierungen von OPC UA in unterschiedlichen Programmiersprachen. Aufgrund der gemeinsamen, standardisierten Basis sind auch unabhängig voneinander entwickelte Clients und Server kompatibel, sofern der OPC UA Standard bei der Programmierung durchgehend befolgt wurde. In der vorliegenden Implementierung wird für den OPC UA Server eine in *C#* geschriebene Serverinstanz von *Unified Automation GmbH* herangezogen. Der verwendete OPC UA Server stellt grundlegende, durch OPC UA vordefinierte Funktionen und Services bereit, wie beispielsweise die erforderlichen Sicherheits- und Verschlüsselungsmechanismen für die Kommunikation mit OPC UA Clients. Für den Import eines Informationsmodells im XML Formal bietet der OPC UA Server die Funktion *ImportUANodeSet*, wodurch die Instanzen bzw. Objekte, Variablen und Methoden, die im vorigen Schritt mittels des Informationsmodells im XML Formal aus dem UA Modeller exportiert wurden, im OPC UA Server erzeugt werden. Bei der initialen Konfiguration der OPC UA Servers wurde ein universeller OPC UA Client verwendet, welcher zur Testzwecken an den OPC UA Server angebunden wurde. Bei dieser Software handelt es sich um *UA Expert* von *Unified Automation GmbH* [151].

Wie in Abbildung 5.8 aufgezeigt, ähnelt *UA Expert* im grundsätzlichen Aufbau der Struktur von UA Modeller. Nach der Herstellung einer Verbindung zum OPC UA Server kann das Informationsmodell mittels einer visuellen Darstellung entlang der Struktur navigiert und die Datenpunkte von Interesse in einen *Data Access View* Bereich per Drag& Drop gezogen werden. Mit Schreibrechten versehene Datenpunkte können manuell verändert werden, um die Funktionsweise des OPC UA Servers zu testen.

Auf die oben beschriebene Weise wurde eine Virtuelle Klemmleiste auf Basis eines OPC UA Servers mit einem importierten Informationsmodell implementiert, die im nachfolgenden Kapitel verwendet wird, um insbesondere die virtuelle Verhaltenslogik in den Digitalen Zwilling zu integrieren.

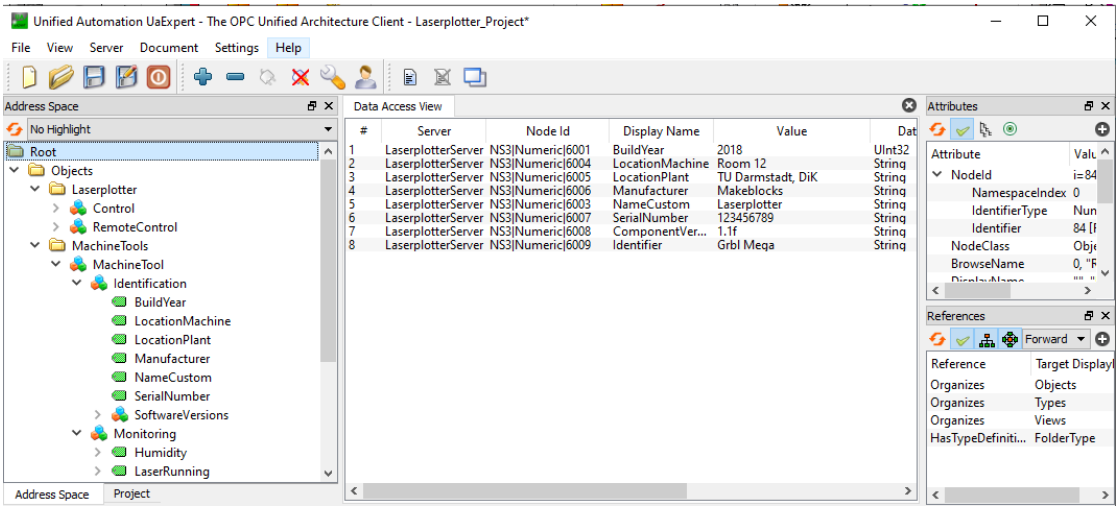

Abbildung 5.8: Universeller OPC UA Client *UA Expert* des Herstellers *Unified Automation GmbH*

5.2 Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik

Nach erfolgreicher Virtualisierung der Verhaltenslogik und Generierung einer *Virtuellen Klemmleiste* wird im folgenden Kapitel die praktische Integration der virtuellen Verhaltenslogik in einen Digitalen Zwilling beschrieben. Dazu wird im ersten Schritt die Architektur des Digitalen Zwillings mit integrierter Verhaltenslogik aufgezeigt. Anschließend wird die Umsetzung der Schnittstellen der einzelnen Partialmodelle erläutert und abschließend die Verbindung des Digitalen Zwillings mit integrierter Verhaltenslogik mit dem Physischen Zwilling dargelegt.

5.2.1 Architektur eines Digitalen Zwillings mit integrierter Verhaltenslogik

Die Architektur des weiterentwickelten Digitalen Zwillings mit dem Fokus auf die *Virtuelle Klemmleiste* ist in der Abbildung 5.5 dargestellt. Der aufgezeigten Architektur ist zu entnehmen, dass die Virtuelle Klemmleiste die Verbindung zwischen dem Physischen und Digitalen Zwilling bildet. Neben einer Möglichkeit Informationsmodelle zu importieren, welche die Schnittstelle definieren, können sowohl physische Komponenten als auch Partialmodelle des Digitalen Zwillings an die Virtuelle Klemmleiste angeschlossen

werden. Seitens des physischen Systems erfolgt die Anbindung in der vorliegenden Implementierung indirekt mittels der Verhaltenslogik des Physischen Zwillings, indem die Steuerungssoftware der Maschine an den OPC UA Server angebunden wird.

Seitens des Digitalen Zwillings erfolgt die Verknüpfung der virtuellen Verhaltenslogik und der Partialmodelle sowohl indirekt, mittels der *Virtuellen Klemmleiste*, als auch direkt zwischen der virtuellen Verhaltenslogik und den Partialmodellen. In den nachfolgenden Kapiteln werden die Implementierungen der einzelnen Verbindungen näher erläutert.

5.2.2 Schnittstellen zwischen Partialmodellen und der Virtuellen Klemmleiste

Zur besseren Übersicht sind die bedeutsamen Stellen der Architektur der *Virtuellen Klemmleiste* bei der Anbindung der Partialmodelle in der Abbildung 5.9 gekennzeichnet und werden gemäß der Nummerierung im Folgenden detailliert erläutert.

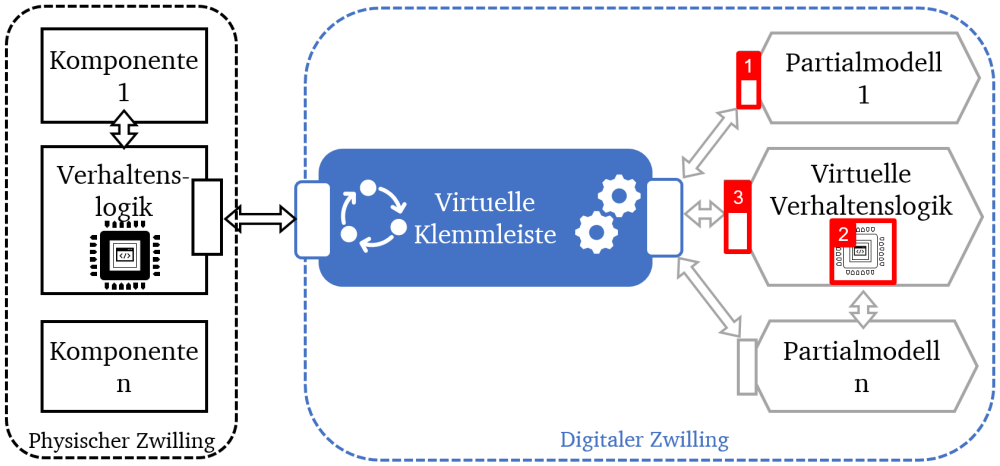

Abbildung 5.9: Betrachtete Elemente der Architektur der Virtuellen Klemmleiste

(1) Schnittstelle des Mechanik-Partialmodells zur Virtuellen Klemmleiste

Die Verknüpfung einzelner Partialmodelle an die mittels OPC UA implementierte Virtuelle Klemmleiste erfolgt mit Hilfe von OPC UA Clients. Im Rahmen der Implementierung wurde exemplarisch eine mechanische Simulation des CNC Laserplotters entwickelt. Die Umsetzung erfolgt mittels des Softwarewerkzeugs *NX - Mechatronic Concept Designer* von *Siemens* [138]. Der *Mechatronic Concept Designer* besitzt einen integrierten OPC UA Client, um sowohl externe Signalquellen einzubinden, als auch Daten aus der Programmumgebung zur Laufzeit einer Simulation zu exportieren.

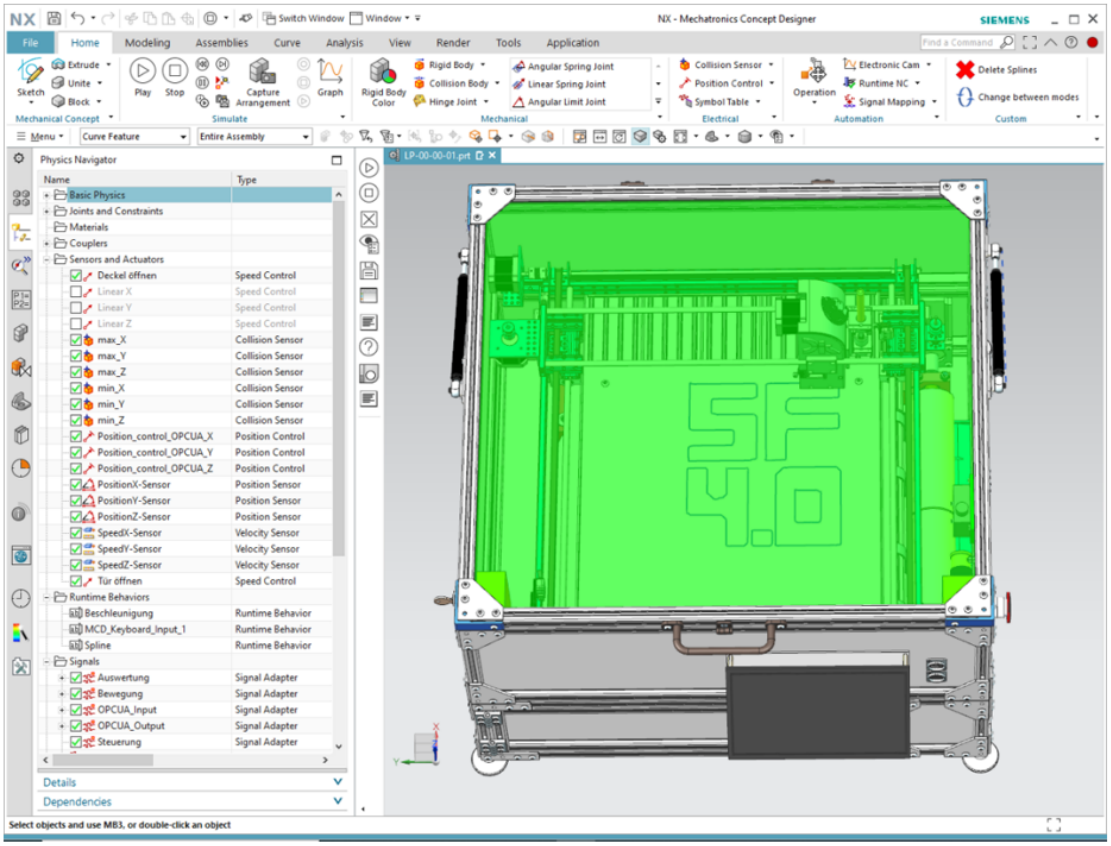

Abbildung 5.10: *NX - Mechatronic Concept Designer* von *Siemens* mit einem Modell des CNC Laserplotters

Ein Ausschnitt aus dem Softwarewerkzeug ist in der Abbildung 5.10 dargestellt. In der Darstellung sind auf der linken Seite die Sensor- und Control-Variablen in einem *Physics Navigator* aufgelistet. Diese sind über eine OPC UA Client Schnittstelle eingebunden und den Elementen der Simulation zugeordnet. Die *Position_control_OPCUA_X* Variable steuert z.B. die Position der X-Achse des simulierten CNC Laserplotters, während der *max_X* Collision Sensor ausgelöst wird, falls der Schlitten der X-Achse einen definierten Bereich (*Collision Box*) überschreitet.

Während der Konfiguration der *Mechatronic Concept Designer* Simulation wird der integrierte OPC UA Client an die Virtuelle Klemmleiste angebunden. Wie in Abbildung 5.11 dargestellt, muss neben der IP-Adresse der *Virtuellen Klemmleiste* auch die Zuordnung der

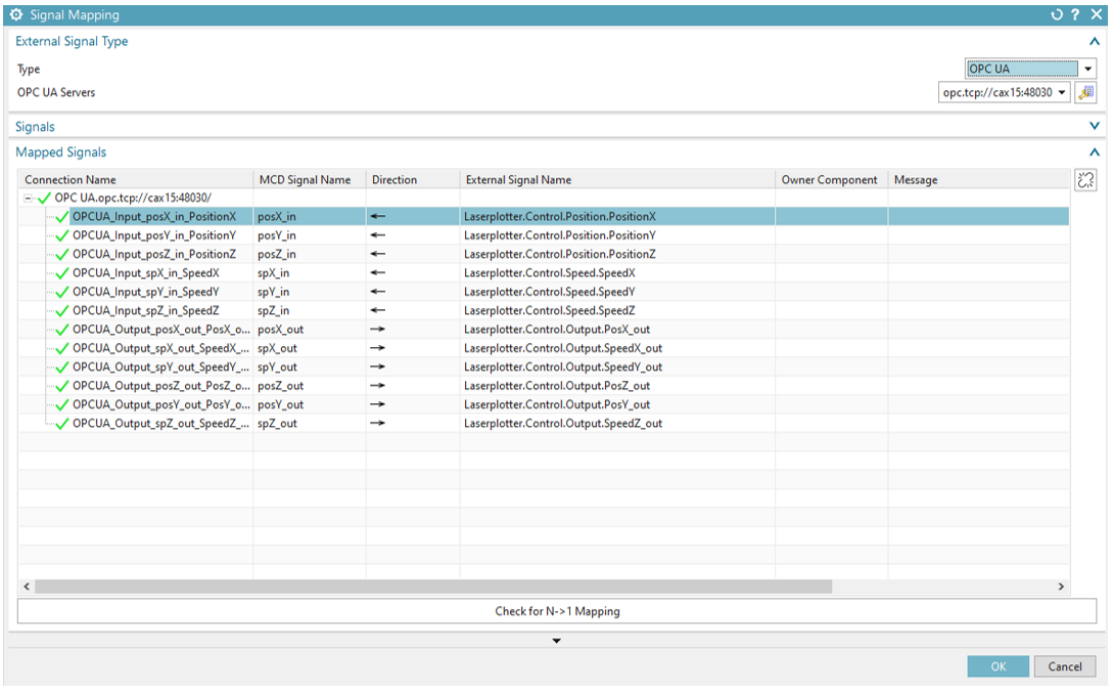

Abbildung 5.11: Signal Mapping des NX - Mechatronic Concept Designer von Siemens

Variablen durchgeführt werden. Dabei werden die im System des *Mechatronic Concept Designers* verwendeten Variablen den korrespondierenden Datenknoten aus dem Informationsmodell des OPC UA Servers zugeordnet. Beispielsweise wird die in der Abbildung 5.11 aufgeführte Variable *posX_in* vom Datenknoten *Laserplotter.Control.Position.PositionX* des verbundenen OPC UA Servers laufend beschrieben.

Auf diese Weise erfolgt die Verbindung des Partialmodells am Beispiel des *Mechatronic Concept Designers* direkt mit der *Virtuellen Klemmleiste*.

(2) Schnittstelle der virtuellen Verhaltenslogik zur Virtuellen Klemmleiste - Monitor-Mikrocontroller

Die beschriebene Integration mittels OPC UA erfordert einen in der Software integrierten OPC UA Client. Dies ist zum aktuellen Stand nur in wenigen Softwarewerkzeugen gegeben, wodurch eine individuelle Lösung in Form eines proprietären OPC UA Clients benötigt wird. Eine solche Lösung wird im Folgenden als ein weiteres Beispiel vorgestellt und

betrifft die virtuelle Verhaltenslogik. Die Schilderungen knüpfen an die Virtualisierung der Verhaltenslogik aus dem Kapitel 5.1 an.

Die verwendete Version 8.10 (*Demonstration*) des Softwarewerkzeugs *Proteus* bietet keine integrierte Funktion, um die Signale der einzelnen GPIOs zu ex- und importieren. Diese Funktion ist jedoch grundlegend, um Partialmodelle des Digitalen Zwillings, die außerhalb von *Proteus* ausgeführt werden, an die virtuelle Verhaltenslogik anzuknüpfen. Folglich wurde diese Funktion implementiert, indem innerhalb von *Proteus* ein als *Monitor-Mikrocontroller* bezeichnetes Modell integriert wurde. Ein Ausschnitt aus dem Monitor-Mikrocontroller ist in der Abbildung 5.12 dargestellt.

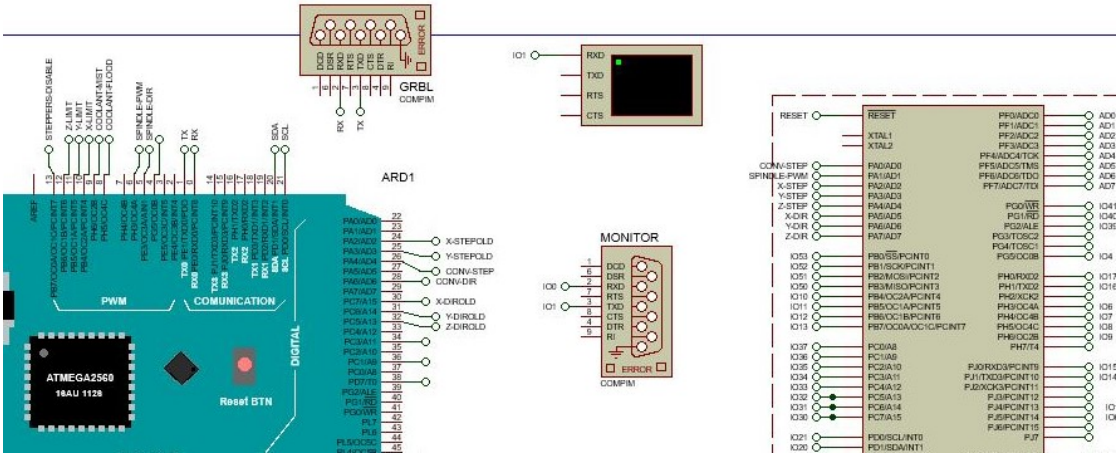

Abbildung 5.12: Ausschnitt der GPIOs des *MEGA 2560* Mikrocontroller Boards (links) und des *Monitor-Mikrocontroller* (rechts, ebenfalls ein *ATMEGA 2560*), welcher die Signale über das *MONITOR-COMPIM* mittels einer COM Verbindung weiterleitet.

Dem Ausschnitt ist zu entnehmen, dass auf den PINs *PA0-PA7* des Monitor-Mikrocontrollers Verbindungen zu den entsprechenden PINs des *MEGA 2560 Bords* hergestellt sind. In dem vorliegenden Beispiel werden folgende Signale mittels des Monitor-Mikrocontrollers aufgenommen:

- *CONV-STEP*: Signale an den Schrittmotor des Papier-Conveyors des CNC Laserplotters,
- *SPINDLE-PWM*: Pulsweitenmodulliertes Signal, welcher zur Steuerung des Lasers verwendet wird,

- *X,Y,Z-STEP*: Signale an die Schrittmotoren der einzelnen Achsen des CNC Laserplotters und
- *X,Y,Z-DIR*: Signal zur Festlegung der Drehrichtung einzelner Schrittmotoren.

Die verknüpften virtuellen Leitungen übertragen die Signale aus dem simulierten Mikrocontroller des Demonstrators in den Monitor-Mikrocontroller.

(3) Schnittstelle der Virtuellen Verhaltenslogik zur Virtuellen Klemmleiste

Die eingebettete Software des Monitor-Mikrocontrollers hat die Aufgabe, die simulierten Signale der virtuellen Elektronik über eine serielle COM Verbindung weiterzuleiten und damit eine Schnittstelle zu *Proteus* herzustellen. Dies erfolgt im Wesentlichen mittels des in Abbildung 5.13 dargestellten C-Codes.

```
48 void setup () {
49 peripheral_setup();
50 Serial.begin(9600);
51 DDRA = B00000000;
52
53 }
54
55 void loop() {
56 peripheral_loop();
57 time = micros();
58 // Wenn PINA &(1<<PAx), wird 1 ausgegeben, wenn der jeweilige Motor aktiv ist, 0, wenn er inaktiv ist
59 sensorStateX = (PINA & (1<<PA2));
60 sensorStateY = (PINA & (1<<PA3));
61 sensorStateZ = (PINA & (1<<PA4));
62 sensorStateSpindle = (PINA & (1<<PA1));
63 sensorStateConv = (PINA & (1<<PA0));
64 // analog zu den Richtungen
65 x_direction = (PINA & (1<<PA5));
66 y_direction = (PINA & (1<<PA6));
67 z_direction = (PINA & (1<<PA7));
68
69 // überprüft, ob der aktuelle Sensorstatus ungleich dem alten Status ist und passt dementsprechend den alten Status an
70 if (sensorStateX != oldStateX || sensorStateY != oldStateY || sensorStateZ != oldStateZ || sensorStateSpindle != oldStateSpindle || sensorStateConv != oldStateConv) {
71 oldStateX = sensorStateX;
72 oldStateY = sensorStateY;
73 oldStateZ = sensorStateZ;
74 oldStateSpindle = sensorStateSpindle;
75 oldStateConv = sensorStateConv;
76   // wenn der aktuelle Status ungleich 0 ist, also aktiv, werden die aktuellen Daten an die serielle Schnittstelle gesendet
77   if (sensorStateX != 0 || sensorStateY != 0 || sensorStateZ != 0 || sensorStateSpindle != 0 || sensorStateConv != 0){
78   counterX++;
79   Serial.println(String(sensorStateX) + String(x_direction) + String(sensorStateY) + String(y_direction) + String(sensorStateZ) +
80                  String(z_direction) + String(sensorStateSpindle) + String(sensorStateConv) + sep + String(time) + sep +String(counterX));
81
82   }
83   }
84 }
85
```

Abbildung 5.13: Ausschnitt aus dem eingebetteten Code des Monitor-Mikrocontrollers.

Bei dem Code handelt es sich um eine Schleife (Code-Zeilen 55-84), welche die Register des Monitor-Mikrocontrollers periodisch ausliest und die aktuellen Zustände (1 oder 0) zu einer Zeichenkette zusammenfügt. In den Code-Zeilen 59-67 wird überprüft, ob der jeweilige PIN aktuell aktiv ist. Anschließend wird in einer *if*-Schleife (Code-Zeilen 70-83) verglichen, ob der aktuelle Zustand dem Zustand aus dem vorherigen Durchgang entspricht oder ob es einen Zustandswechsel gab. Im Falle einer Zustandsänderung wird

in den Code-Zeilen 77-82 eine serielle Nachricht zusammengestellt und über eine serielle Verbindung versendet. Der vorgestellte Code wurde hinsichtlich Geschwindigkeit optimiert, da die Schleife sehr hohe Taktraten erreichen muss.

Nach der Verarbeitung durch den Monitor-Mikrocontroller werden die Daten über eine serielle Verbindung exportiert. Hierfür wurde eine *MONITOR-COMPIM* Verbindung eingerichtet, siehe Abbildung 5.12. Die *TX/RX* Kanäle des Monitor-Mikrocontrollers werden an die MONITOR-COMPIM angebunden und durch Konfiguration an ein COM-Port des Simulationsrechners weitergeleitet. Diese MONITOR-Schnittstelle ermöglicht es dadurch, externe Simulationswerkzeuge über die COM-Verbindung zu integrieren.

5.2.3 Verbindung mit dem Physischen Zwilling

Im nächsten Schritt erfolgt die Anbindung des Physischen Zwillings an die Virtuelle Klemmleiste. Dies erfolgt ebenfalls mittels OPC UA, indem das physische System durch einen in der Steuerungssoftware integrierten OPC UA Client angebunden wird. Die Steuerungssoftware des CNC Laserplotters wird zu diesem Zweck mit einem Client erweitert, welcher in der Lage ist, die Steuerbefehle und Sensordaten sowohl an die Virtuelle Klemmleiste umzuleiten, als auch Daten zu empfangen. In dieser Konstellation ist es möglich, sowohl die Partialmodelle des Digitalen Zwillings mittels der physischen Verhaltenslogik zu steuern, als auch den Digitalen Zwilling zur Steuerung des physischen Systems zu nutzen. Diese Anwendungsfälle sind bedeutsam für die Umsetzung der in Kapitel 3.3 beschriebenen Anwendungsfälle und werden im Rahmen der Validierung weiter erläutert.

5.3 Fazit

Das Kapitel beinhaltet die prototypische Implementierung des entwickelten Konzepts einer Virtuellen Klemmleiste zur Integration der Verhaltenslogik in den Digitalen Zwilling eines mechatronischen Systems. Zu diesem Zweck erfolgt im ersten Schritt die prototypische Virtualisierung der Elektronik eines CNC Laserplotters mithilfe von *Proteus*, einem Modellierungs- und Simulationswerkzeug für elektronische Komponenten. In der virtualisierten Elektronik kann anschließend die eingebettete Software des physischen Systems integriert und in der Simulationsumgebung ausgeführt werden. Dies erzeugt die virtuelle Verhaltenslogik des Digitalen Zwillings. Als Anknüpfungspunkt der virtuellen Verhaltenslogik dient eine Virtuelle Klemmleiste, welche die OPC UA Technologie instrumentalisiert und mittels eines Informationsmodells die Schnittstelle definiert. An der *Virtuellen Klemmleiste* können anschließend sowohl physische Komponenten als auch Partialmodelle angebunden werden. Exemplarisch sind zwei Beispiele aufgezeigt. Erstens

wird ein Partialmodell in Form einer mechanischen Simulation mittels des *Mechatronic Concept Designers* umgesetzt und angebunden. In einem zweiten Beispiel wird zur Anbindung der virtuellen Verhaltenslogik die Mikrocontroller Simulation durch eine proprietäre Schnittstelle erweitert. Zuletzt erfolgt die Beschreibung der Verbindung zwischen dem Physischen- und dem Digitalen Zwilling.

Im nachfolgenden Kapitel wird die beschriebene Implementierung genutzt, um das entwickelte Konzept am Beispiel von konkreten Anwendungsfällen zu validieren.

6 Validierung und Verifikation

Im nachfolgenden Kapitel wird das im Rahmen der Dissertation entwickelte Konzept hinsichtlich seiner Tragfähigkeit untersucht. Dafür wird die Implementierung am Beispiel des CNC Laserplotters herangezogen und analysiert, inwiefern die im Kapitel 3.3 vorgestellten Anwendungsfälle erfüllt und ermöglicht werden. Dazu wird anhand der Anwendungsfälle *Integration des Verhaltens in den Digitalen Zwilling* und *Steuerung des Digitalen Zwillings* das Konzept und die Implementierung validiert. Anschließend erfolgt eine Verifikation der gesamten Anforderungsprofils hinsichtlich des Erfüllungsgrades.

6.1 Anwendungsfall - Integration des Verhaltens in den Digitalen Zwilling

Der in Kapitel 3.3 vorgestellte Anwendungsfall beschreibt die Integration der Verhaltenslogik in den Digitalen Zwilling und ist in der Abbildung 3.1 mittels eines UML-Anwendungsfalldiagramms visualisiert. Im Folgenden werden nur die wesentlichen Bestandteile des Anwendungsfalls, welche auf den Erläuterungen im Kapitel 5 Implementierung aufbauen, vorgestellt.

Die Akteure *Softwareingenieur*, *Elektronikingenieur*, *Systemingenieur*, *Konstrukteur/Simulationsingenieur* generieren unter Zuhilfenahme der *Ressource Datenbank für Anforderungen* den Teil des Digitalen Zwillings, welcher für die originalgetreue Abbildung des Verhaltens erforderlich ist. Dabei wird eine Schnittstelle generiert, an der sowohl die virtuelle Abbildung der Elektronik, als auch mechanische Simulationen anknüpfen können.

Der vorgestellte Anwendungsfall wurde auf Basis des CNC Laserplotters durchgeführt und zeigt die Möglichkeiten und die offenen Potenziale des Konzepts und der Implementierung auf. Eine erste Herausforderung besteht darin, dass für diese komplexe Aufgaben diverse Expertisen notwendig sind. Durch den Fokus auf das System des CNC Laserplotters, welches eigens weiterentwickelt und im Rahmen der Forschungstätigkeiten intensiv untersucht wurde, war es für den Laserplotter möglich, die erforderlichen Fachkenntnisse zusammenzutragen. Demnach wurde gemäß dem im Konzeptteil beschriebenen Teilprozess *A11* die eingebettete Software *Grbl-Mega* analysiert. Zusammen mit den Modellen

der virtualisierten Elektronik konnte eine virtuelle Verhaltenslogik erschaffen werden. Diese wird an die anschließend generierte Virtuelle Klemmleiste angebunden und mit Modellen der Mechanik des Laserplotters verknüpft.

Der durchgeführte Prozess, mit dem Fokus auf die Trennung der einzelnen Domäne in Form eines Aktivitätsdiagramms, ist in der Abbildung 6.1 dargestellt.

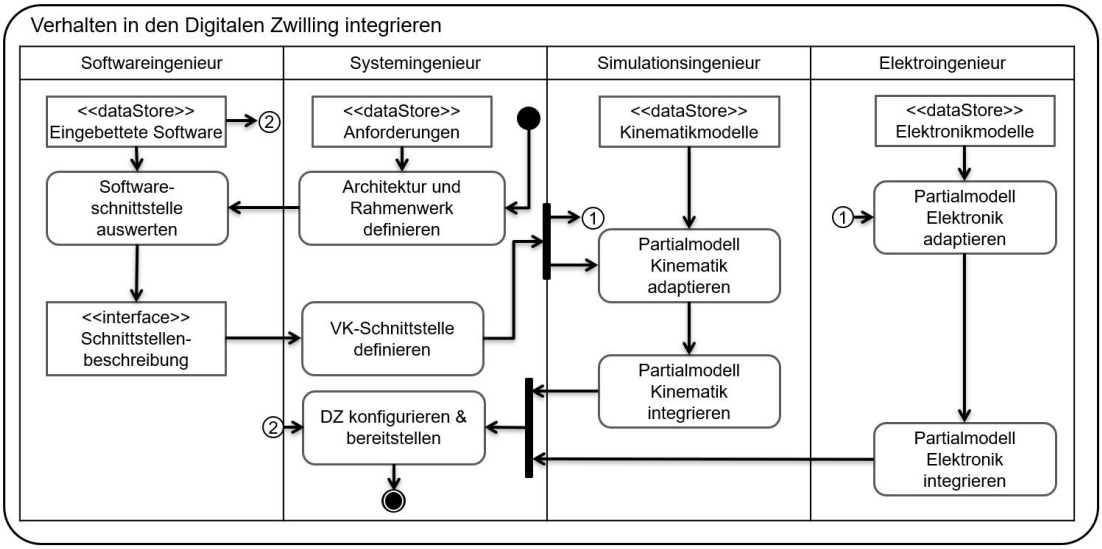

Abbildung 6.1: Prozess der Integration des Verhaltens in den Digitalen Zwilling mit dem Fokus auf das Zusammenwirken der beteiligten Disziplinen.

Der Beschreibung ist zu entnehmen, dass jeder Domäne zur Durchführung des Anwendungsfalls eine Datenbasis *«dataStore»* als Grundlage bereitsteht. Die Daten stammen aus der Produktentwicklung des Systems und bilden die Voraussetzungen für die erfolgreiche Realisierung des Anwendungsfalls. Demnach ist gemäß Beschreibung des Produktentwicklungsprozesses anhand des V-Modells aus dem Kapitel 2.1 die Umsetzung des Anwendungsfalls erst nach der Implementierung der Systemelemente möglich.

Bei der Durchführung des Anwendungsfalls zeigten sich einige Besonderheiten, die im Folgenden erläutert werden. Wie bereits geschrieben, wird zur Auswertung der eingebetteten Software der Quellcode oder eine adäquate Dokumentation benötigt, um den Zusammenhang zwischen der Verarbeitung in der Software und der Elektronik sowie den mechanischen Teilen des Digitalen Zwillings herstellen zu können. Es hat sich gezeigt, dass die technische Dokumentation der verwendeten Software nicht alle notwendigen Informationen enthält, sondern sich mehr auf die Einrichtung und Konfiguration der Soft-

ware fokussiert. Durch die Verwendung quell-offener Software war die direkte Auswertung der eingebetteten Software möglich, womit die erforderlichen Informationen ermittelt werden konnten. In der Abbildung 6.2 ist ein beispielhafter Ausschnitt aus der *cpu_map* Header-Datei von *Grbl-Mega* abgebildet und wird im Folgenden exemplarisch erläutert.

```
29  #ifdef CPU_MAP_2560_INITIAL // (Arduino Mega 2560) Working @EliteEng
30
31    // Serial port interrupt vectors
32    #define SERIAL_RX USART0_RX_vect
33    #define SERIAL_UDRE USART0_UDRE_vect
34
35    // Define step pulse output pins. NOTE: All step bit pins must be on the same port.
36    #define STEP_DDR      DDRA
37    #define STEP_PORT     PORTA
38    #define STEP_PIN      PINA
39    #define X_STEP_BIT    2 // MEGA2560 Digital Pin 24
40    #define Y_STEP_BIT    3 // MEGA2560 Digital Pin 25
41    #define Z_STEP_BIT    4 // MEGA2560 Digital Pin 26
42    #define STEP_MASK ((1<<X_STEP_BIT)|(1<<Y_STEP_BIT)|(1<<Z_STEP_BIT)) // All step bits
```

Abbildung 6.2: Ausschnitt aus *cpu_map.h* des *Grbl-Mega* mit Darstellung der Zuordnung von Variablen zu entsprechenden Registern, PORTs und PINs

Bei der Auswertung der eingebetteten Software muss unter anderem analysiert werden, auf welchen Registern, PORTs und PINs des Mikrocontrollers (*ATMega 2560*) und des Mikrocontroller-Boards (*Arduino Mega 2560*) welche Signale vorliegen. In der Zeile 39 in der Abbildung 6.2 ist beispielsweise angegeben, dass die Schritt-Signale zur Steuerung des Motors der X-Achse über den *PIN 2* des Mikrocontrollers bzw. *PIN 24* des Mikrocontroller-Boards (siehe auch Abbildung 5.12) ausgegeben werden. Diese Informationen müssen im anschließenden Schritt der Konfiguration der eingebetteten Elektronik berücksichtigt werden.

Eine weitere Besonderheit bei der Durchführung des Anwendungsfalls zeigt sich bei der Virtualisierung der Elektronik des CNC Laserplotters. Die im Laserplotter verwendeten elektronischen Komponenten sind Standardbausteine, die in unterschiedlichen Systemen eingesetzt werden können. Trotz der weiten Verbreitung und teilweisen Offenheit[1] der Komponenten existieren keine geeigneten digitalen Modelle. Nur die Unternehmen, welche diese Komponenten entwickeln, besitzen detaillierte virtuelle Modelle und spezialisierte Entwicklungsumgebungen, welche die Funktionsweise der Elektronik in einer Simulation abbilden können. Aus diesen Gründen muss ein vereinfachtes Modell der eingebetteten

[1] Die Baupläne einiger Komponenten, wie beispielsweise des *RAMPS 1.4 - Boards*, sind frei verfügbar und können für die Analyse und Herstellung dieser Komponenten verwendet werden.

Elektronik entwickelt werden, um den Anwendungsfall durchführen zu können.

Zusammenfassend konnte der beschriebene Anwendungsfall der Integration der Verhaltenslogik in einen Digitalen Zwilling am Beispiel des CNC Laserplotters erfolgreich durchgeführt werden. Die Umsetzung lieferte zusätzliche Erkenntnisse hinsichtlich der Anwendbarkeit des Konzepts und dem Einsatz der Implementierung. Weiterführende Bewertung dieser Erkenntnisse im Rahmen der Validierung folgt im Kapitel 6.3.

6.2 Anwendungsfall - Steuerung des Digitalen Zwillings

Der zweite Anwendungsfall beschreibt die Steuerung des Digitalen Zwillings und basiert auf der erfolgreichen Durchführung des ersten Anwendungsfalls. Dabei soll ermöglicht werden, dass der Nutzer auf die gleiche Weise mit dem Digitalen Zwilling interagieren kann, wie es bereits mit dem physischen System möglich ist. Das UML Diagramm in Abbildung 3.2 stellt den Anwendungsfall dar. Der Anwendungsfall ist in der Nutzungsphase zu verorten und wird von einem *Nutzer*-Akteur durchgeführt. Der Nutzer kann nach einer Konfiguration des Digitalen Zwillings diesen mittels der gleichen Schnittstelle steuern, wie er es vom physischen System gewohnt ist. Dabei kommt die Virtuelle Klemmleiste zum Einsatz, welche die Verbindung zwischen den Elementen des Digitalen Zwillings und dem physischen System herstellt. Als externe Ressourcen können Datenbanken für Modelle des Digitalen Zwillings und Simulationsserver optional auftreten. Ein Physischer Zwilling ist gemäß der Definition eines Digitalen Zwillings stets erforderlich. Im vorliegenden Anwendungsfall ist es jedoch notwendig und zweckmäßig, die bidirektionale Verbindung zwischen dem Digitalen und Physischen Zwilling zeitweise trennen zu können, um beispielsweise isolierte Tests durchführen zu können.

Im Folgenden werden die Besonderheiten der Umsetzung des Anwendungsfalls beschrieben. Zur Durchführung des Anwendungsfalls werden zunächst zusätzliche Programme implementiert, welche insbesondere die Steuerung des Digitalen Zwillings ermöglichen. Dazu wird die Benutzungsoberfläche *bCNC* des physischen Systems (siehe Abbildung 6.3) stattdessen an den Digitalen Zwilling angebunden.

Zur Steuerung des Digitalen Zwillings wird *bCNC* mit der virtuellen Verhaltenslogik verbunden. Dazu musste die quell-offene Software angepasst werden, indem ein OPC UA Client integriert wurde. Zur Verdeutlichung des Umfangs einer Erweiterung eines Programms mit einem OPC UA Client, sind in der Abbildung 6.4 die Softwarecodepassagen dargestellt.

Der Implementierung des OPC UA Clients ist zu entnehmen, dass neben dem Import der OPC UA Bibliotheken für Python (Zeilen 73-75), eine *try*-Schleife (Zeilen 127-139) für die Herstellung der Verbindung zum OPC UA Server unter der lokalen IP-PORT Adresse

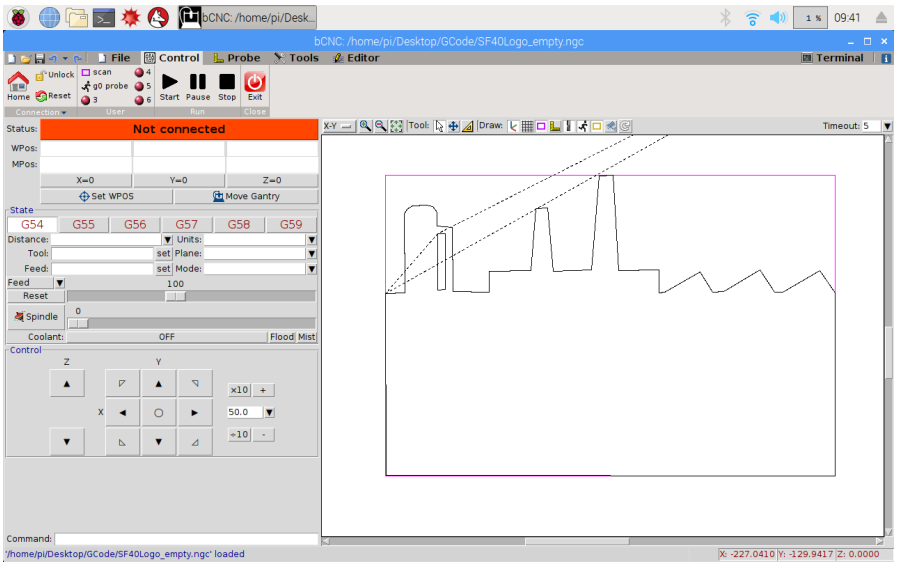

Abbildung 6.3: bCNC als Benutzungsoberfläche sowohl für das physische System als auch für den Digitalen Zwilling

opc.tcp://192.168.1.10:48030/ implementiert wurde. Zur Ausführung der implementierten Funktion wurden zusätzliche Bedienknöpfe in der Benutzungsoberfläche *bCNC* eingefügt (Zeilen 316-317). Die restliche Implementierung der *remoteControl*- und *quitOPCUA*-Funktion befindet sich im Anhang 9.1.

Durch die dargestellte Erweiterung bietet *bCNC* die Möglichkeit, zwischen der Steuerung des physischen Systems und des Digitalen Zwillings umzuschalten. Damit unterschied sich die Nutzung von *bCNC* zur Steuerung des Digitalen Zwillings nicht von der gewohnten Art des Umgangs mit dem physischen System. Der Abbildung 6.5 ist zu entnehmen, dass die Bearbeitung des gleichen G-Codes in einem ähnlichen Plotergebnis resultiert. Kommunikationspartner wie beispielsweise *bCNC* erkennen keinen Unterschied, ob es sich um das physische System oder um den Digitalen Zwilling dessen handelt. Der beschriebene Anwendungsfall der Steuerung des Digitalen Zwillings konnte somit erfolgreich durchgeführt werden.

In Abbildung 6.6 ist der Anwendungsfall in einem Sequenzdiagramm dargestellt. In der Visualisierung stellt der Nutzer mittels der Benutzungsoberfläche *bCNC* zunächst mittels der beschriebenen Funktion *remoteControl()* eine Verbindung zur *Virtuellen Klemmleiste* her. Anschließend kann der Nutzer durch Betätigung entsprechender Steuerelemente der Benutzungsoberfläche die Steuerung ausführen. Die Benutzungsoberfläche generiert dabei

```
73  from opcua import Client
74  from opcua import ua
75  from opcua import crypto
```

```
127         try:
128             self.client=Client("opc.tcp://192.168.1.10:48030/")
129             print("Client started")
130 
131             self.client.connect()
132             self.client.activate_session()
133             print("OPCUA connected")
134             print(self.client)
135         except:
136             print("OPCUA Server not working")
137             print("No OPCUA connection established")
138             self.client=None
139             pass
```

```
316         self.bind('<<RemoteControl>>',  lambda e,s=self: s.remoteControl(self.client))
317         self.bind('<<QuitOPCUA>>',  lambda e,s=self: s.quitOPCUA(self.client))
```

Abbildung 6.4: Erweiterung der *bCNC* Software um einen OPC UA Client in der *__main__.py*

einen Steuerbefehl und versendet diesen an die Virtuelle Klemmleiste, die den Befehl speichert. Der Digitale Zwilling liest in einem definierten Zyklus den vorgegeben Soll-Zustand in der *Virtuellen Klemmleiste* aus und aktualisiert bei Veränderungen entsprechend den eigenen Zustand. Der neue Ist-Zustand des Digitalen Zwillings wird anschließend durch die virtuellen Sensoren erfasst und der Ist-Zustand wird an die Schnittstelle übermittelt. Der Nutzer hat nur die Möglichkeit, den aktualisierten Ist-Zustand des Digitalen Zwillings zu überprüfen. Die Steuerung des Digitalen Zwillings wird durch die Initiierung der *quitOPCUA()* Funktion beendet und die Verbindung getrennt.

Der vorgestellte Anwendungsfall bildet seinerseits die Grundlage für weitere Anwendungsfälle des Digitalen Zwillings. So können beispielsweise Veränderungen oder Erweiterungen des physischen Systems und dessen eingebetteter Software zunächst vollständig und originalgetreu mittels des Digitalen Zwillings getestet werden. Der Umfang der virtuell abgebildeten Komponenten kann dabei skalieren, indem die Modelle entsprechend ausgewählt werden. Es kann ausreichen, nur die Bewegungen mechanischer Komponenten mittels geeigneter Modelle, wie z.B. die beschriebene Mehrkörpersimulation in *Siemens Mechatronic Concept Designer*, zu analysieren. Es ist jedoch auch möglich detaillierte Simulationsmodelle der internen Abläufe und Verhalten der Aktoren und Sensoren mit einzubinden.

Mit dem vorgestellten Anwendungsfall konnte gezeigt werden, dass es durch die Virtuelle Verhaltenslogik möglich wird, den Digitalen Zwilling auf die gleiche Weise und mit den gleichen Werkzeugen zu steuern, wie bereits das physische System. Die kritische Bewertung der Anwendungsfälle und der Implementierung folgt im nächsten Kapitel.

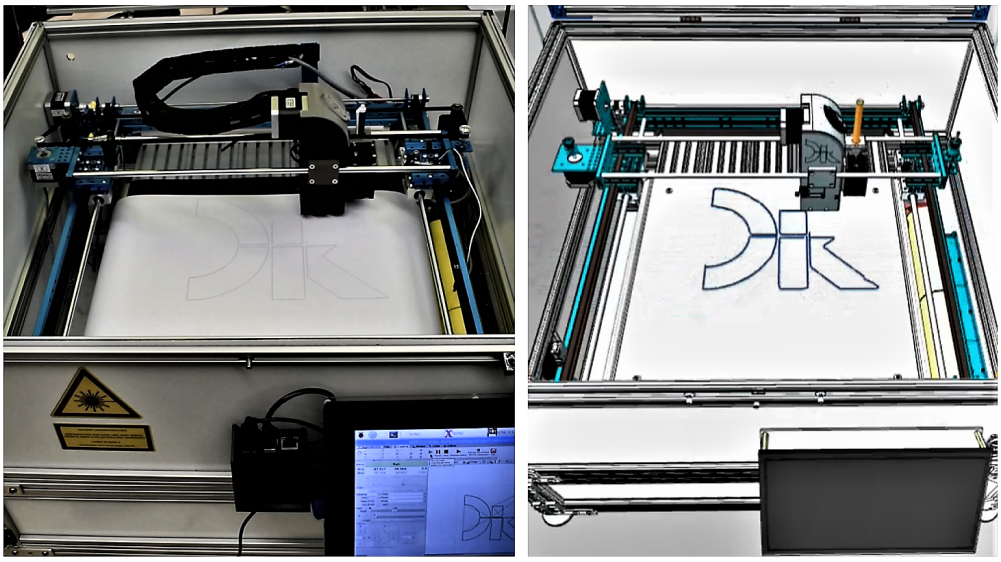

Abbildung 6.5: Validierung des Verhaltens des Digitalen Zwillings durch Vergleich mit dem physischen System

6.3 Bewertung der Validierung

Das entwickelte Konzept und die vorgestellte Implementierung konnten eingesetzt werden, um sowohl den Anwendungsfall *Integration des Verhaltens in den Digitalen Zwilling*, als die *Steuerung des Digitalen Zwillings* durchzuführen. Damit wurde die Eignung des Konzepts und der Implementierung untersucht und konnte erfolgreich validiert werden.

Das Validierungsbeispiel *Integration des Verhaltens in den Digitalen Zwilling* hat die Notwendigkeit von enger Zusammenarbeit der einzelnen Disziplinen der Informatik, Elektrotechnik und Maschinenbau deutlich bestätigt. Die Umsetzung eines Digitalen Zwilling mit integrierter Verhaltenslogik erfordert detailliertes Fachwissen aus den Disziplinen, um ein realitätsnahes und originalgetreues Abbild zu erschaffen. Dies hat die anfangs getroffene Annahme, dass die Entwicklung des Digitalen Zwillings mit integrierter Verhaltenslogik zusammen mit der Produktentwicklung des physischen Systems stattfinden sollte, deutlich belegt.

Durch das Zusammenwirken der Disziplinen in einem gemeinsamen System, in dem die einzelnen Komponenten und Partialmodelle miteinander vernetzt sind, wird die Notwendigkeit deutlich, standardisierte interne Schnittstellen zu verwenden. Diese wesentliche Charakteristik des Digitalen Zwillings wird durch den Einsatz der OPC UA Technolo-

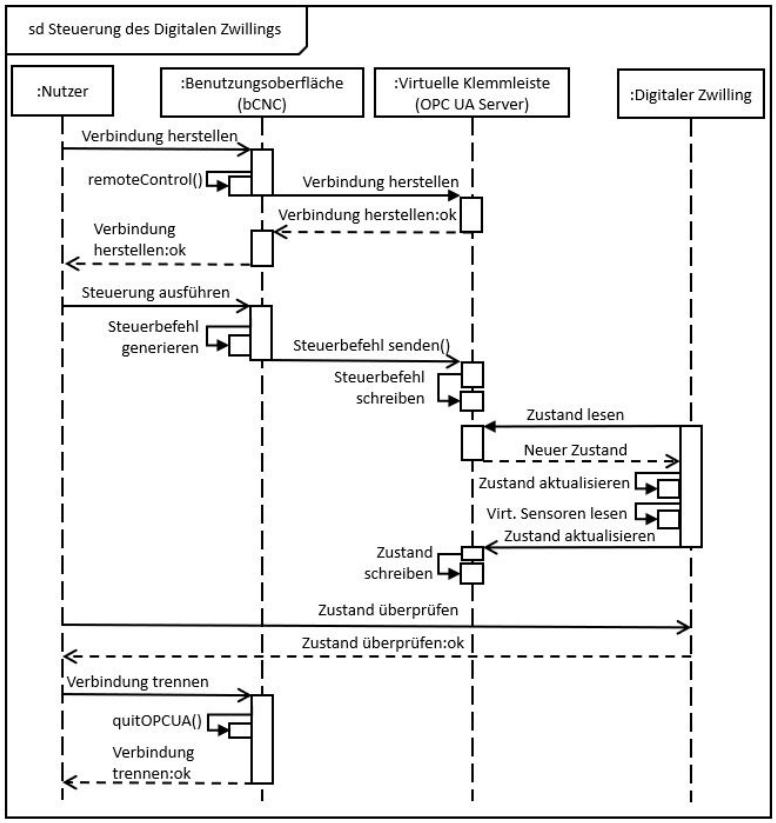

Abbildung 6.6: Sequenzdiagramm der Steuerung des Digitalen Zwillings durch den Nutzer

gie zur Umsetzung der *Virtuellen Klemmleiste* verwirklicht. Damit wurde nicht nur die Unabhängigkeit des Konzepts von einzelnen Herstellern und Entwicklungsumgebungen sichergestellt, sondern auch eine gemeinsame Sprache der Disziplinen gewählt. Durch die Entwicklung der Companion Specifications in diversen Branchen ist somit zu erwarten, dass diese Technologie eine geeignete Schnittstelle für den Digitalen Zwilling ist. Zum aktuellen Zeitpunkt finden sich jedoch Lücken sowohl hinsichtlich Branchen als auch hinsichtlich Unterstützung durch Entwicklungswerkzeuge. Insbesondere in der Disziplin der Elektronik ist OPC UA wenig verbreitet, weshalb die Abbildung der Elektronik im Digitalen Zwilling sich als schwierig, nur über Umwege möglich, erwiesen hat.

Die Validierung hat weiterhin gezeigt, dass die bidirektionale Verbindung zwar möglich ist, jedoch weiterer Forschung bedarf. Das Konzept und die Implementierung ermög-

licht die Verbindung in beide Richtungen, die jedoch sinnvoll konfiguriert werden muss. Demnach sind weiterführende Entwicklungen notwendig, um eine sichere gegenseitige Beeinflussung zu ermöglichen. Zum aktuellen Zeitpunkt sind, infolge der im Konzept beschriebenen Zirkularität des Zwillingssystems, unerwünschte Wechselwirkung mit unerwarteten Folgen möglich. Aus diesem Grund ist eine frühe Festlegung der Parameter und Variablen der Wechselwirkung notwendig und muss weiter erforscht werden.

Eine weitere Eigenschaft, welche im Rahmen der Validierung bedeutsam war, ist die Echtzeitfähigkeit des Zwillingssystems. Sowohl die Vorhersagbarkeit, als auch die Geschwindigkeit der Kommunikation zwischen den Zwillingen kann nicht mit ausreichender Sicherheit garantiert werden. Dies ist zum Teil der prototypischen Implementierung geschuldet, welche noch weiteren Optimierungspotenzial besitzt, indem beispielsweise *OPC UA over TSN* umgesetzt wird. Zum großen Teil resultieren die Verzögerungen jedoch aus der Tatsache, dass die Verarbeitung der Daten im Digitalen Zwilling sehr umfangreich werden kann. Die verwendeten Simulationen zur Abbildung bestimmter Eigenschaften des Digitalen Zwillings benötigen Zeit und hängen von Einflussfaktoren ab. Zu diesen Faktoren gehört z.B. die Leistungsfähigkeit der Plattformen und der verwendeten Kommunikationstechnologien und Datenbanken, die an der Ausführung des Digitalen Zwillings beteiligt sind. Aus heutiger Sicht kann der Digitale Zwilling nur für zeit-unkritische Anwendungsfälle eingesetzt werden und ist im Kontext der industriellen Produktion demnach stark eingeschränkt.

Ein weiterer Diskussionspunkt im Rahmen der Validierung ist die Verwendung unterschiedlicher Simulationsarten im Digitalen Zwilling. Wie im Kapitel 2.4 beschrieben, können Simulationen grundsätzlich in *diskret* und *kontinuierlich* unterteilt werden. Im Digitalen Zwilling kommen beide Simulationsarten zum Einsatz und verstärken die Problematik der Kompatibilität der einzelnen Partialmodelle, die miteinander wechselwirken müssen, um das physische System weitgehend abzubilden. Die im Rahmen der Implementierung generierte Simulationskette einhält sowohl diskrete als auch kontinuierliche Simulationen. Während die digitalen Signale des Mikrocontrollers einen diskreten Verlauf (1 oder 0) aufweisen, wird ein mechanisches Partialmodell kontinuierlich simuliert. Demnach müssen in betroffenen Anwendungsfällen sogenannte *Wrapper* entwickelt werden, um die unterschiedlichen Simulationen miteinander verbinden zu können. Im Rahmen der Validierung wurde beispielsweise in der Matlab/Simulink-Umgebung die Überführung zwischen diskreten Signalen des Mikrocontrollers und der kontinuierliche Drehung der Schrittmotoren durch Interpolation erreicht. Weitere Erforschung der Herausforderungen hybrider Co-Simulationen wurde im Rahmen der Dissertation jedoch nicht verfolgt und besitzt Potenzial für weitere Untersuchungen.

Die Validierung eröffnete weitere Herausforderungen hinsichtlich der Annahmen, unter denen das Konzept und die Implementierung entwickelt wurden. Dies betrifft insbesonde-

re die Aufrüstung eines bestehenden Systems ohne direkte Beteiligung des ursprünglichen Herstellers. In der Anwendung des Konzepts in Zusammenhang mit industrieller eingebetteter Software müssen zunächst rechtliche Rahmenbedingungen geschafften werden, um die eingebettete Software im Digitalen Zwilling nutzen zu können. Die in der Praxis bereitgestellte technische Dokumentation beschreibt in der Regel die Verwendung und Konfiguration der Software und enthält wenig Hinweise bezüglich der internen Funktionsweise. Genaue Angaben zum internen Aufbau würden wesentliche Teile des Know-Hows der eingebetteten Software offenlegen. Ebenso wenig ist im Kontext kommerzieller Software der Quellcode bekannt. Aus diesen Gründen sind neuartige vertragliche Rahmenbedingungen und Geschäftsmodelle notwendig, um in der Praxis die Informationen und Ressourcen zu erhalten, welche für die Integration der Verhaltenslogik im Digitalen Zwilling notwendig sind. Solange diese Art von rechtlichen Voraussetzungen nicht gegeben sind, ist die Entwicklung und Bereitstellung eines Digitalen Zwillings durch den Hersteller des physischen Systems zu bevorzugen.

Zusammenfassend hat die Validierung gezeigt, dass das entwickelte Konzept einen weiteren Schritt in Richtung eines umfangreichen Digitalen Zwilling ermöglicht. Gleichzeitig wurden einige oben beschriebene Herausforderungen offengelegt, die durch weitere Forschung gelöst werden müssen, bis tatsächlich ein Digitaler Zwilling entwickelt wird, welcher den *Grieves' Test of Virtuality* (vgl. Kapitel 2.3.5) bestehen kann. Während ein *Visueller Test* eines Objekts mit den modernen Werkzeugen durchaus zu erreichen ist, nimmt die Abbildung des Verhaltens in einem *Performanztest* und einen *Reflektivitätstest* eine führende Rolle ein. Das entwickelte Konzept trägt somit zum Erreichen von diesen fortgeschrittenen Stufen des Digitalen Zwillings maßgeblich bei.

6.4 Verifikation des Anforderungsprofils

Die Durchführung der Validierung anhand der beiden Anwendungsfälle hat die Tragfähigkeit und Anwendbarkeit des Konzepts einer Virtuellen Klemmleiste zur Integration der Verhaltenslogik in den Digitalen Zwilling demonstriert. Im Folgenden wird im Rahmen einer Verifikation überprüft, inwiefern die im Kapitel 3.4 aufgestellten Anforderungen erfüllt werden.

Die Anforderungen 1 bis 3 an die eingebettete Software sind vollständig erfüllt. Die Möglichkeit, die gleiche eingebettete Software im Digitalen Zwilling verwenden zu können bildete eine Hauptmotivation des Konzepts. Diese Anforderungen konnten erfüllt werden, indem eine Simulation der eingebetteten Hardware eingesetzt wurde, in welcher anschließend die eingebettete Software unverändert ausgeführt werden konnte. Durch die Verwendung der Entwicklungs- und Simulationsumgebung *Proteus* waren keine

Anpassungen in der eingebetteten Software notwendig.

Außer der nur teil-erfüllten Festforderung 8 sind die Anforderungen an die Virtuelle Klemmleiste als die virtuelle Software-Hardware Schnittstelle vollständig erfüllt. Die Virtuelle Klemmleiste dient als Schnittstelle zwischen der eingebetteten Software und den davon abhängigen Modulen des Digitalen Zwillings (Anforderung 4) und basiert auf einem Informationsmodell, welches einerseits durch die Verwendung von OPC UA Companion Specifications bereits weitgehend definiert ist und andererseits beliebig erweitert werden kann. Dadurch können erforderlichen Daten in dem Informationsmodell der Schnittstelle eingebettet werden (Forderung 5). Es ist möglich, sowohl bei Neuentwicklung, als auch bei Aufrüstung eine Virtuelle Klemmleiste zu generieren (Anforderung 6). Gemäß den Erläuterungen im Kapitel 4.3.3 und 5 kann ein Informationsmodell mittels XML in einen OPC UA Server importiert werden oder das Informationsmodell wird in einem vorgelagerten Schritt mittels eines unterstützenden Werkzeuges, wie z.B. *UAModeller*, erstellt bzw. erweitert (Anforderung 7). Die Virtuelle Klemmleiste kann grundsätzlich domänenübergreifend eingesetzt werden. Die gewählte Technologie OPC UA ist bereits in einigen Branchen bekannt und verbreitet. Es hat sich jedoch gezeigt, dass die Unterstützung dieser Technologie sich bisher auf kleinere Anzahl an Softwarewerkzeugen begrenzt. In einigen Domänen, wie beispielsweise der Herstellung von elektronischen Komponenten, wird OPC UA dagegen noch nicht häufig eingesetzt. Aus diesem Grund wurden während der Implementierung individuelle Lösungen gefunden, um diese Werkzeuge dennoch an die Virtuelle Klemmleiste anbinden zu können. Die Anforderung 8 wird aus diesen Gründen als nur teilweise erfüllt betrachtet. Für die Erstellung und Erweiterung der *Virtuellen Klemmleiste* bzw. des zugrundeliegenden Informationsmodells wurden sowohl eine methodische Vorgehensweise (Anforderung 9) als auch ein unterstützendes Werkzeug (Anforderung 10) vorgestellt.

Die Anforderungen an den Digitalen Zwilling konnten mindestens teil-erfüllt werden. Der Physische und der Digitale Zwilling können mittels der Virtuelle Klemmleiste bidirektional verbunden werden (Anforderung 11). Die implementierten Teile des Digitalen Zwillings sind jedoch teilweise auf domänenspezifische Autorenwerkzeuge angewiesen, weshalb die Anforderung 12 nur teilweise erfüllt werden konnte. Insbesondere ist dabei die Modellierungs- und Simulationssoftware *Proteus* [87] zu nennen. Die verwendete Version unterstützt nicht den Export von Simulationsmodellen oder Solvern, wie dies mittels der FMI Technologie möglich wäre. Der Versuch andere Werkzeuge, wie beispielsweise *SimulAVR* [149], zu verwenden führten zu keinem geeigneten Ergebnis, da bei diesem Werkzeug einerseits der verwendete Mikrocontroller nicht unterstützt wurde und andererseits keine weiteren Modelle von elektronischen Komponenten integrierbar sind. Sowohl die konzeptionelle Ausgestaltung als auch die Implementierung wurden darauf ausgelegt, dass der Digitale Zwilling auch in der Nutzungsphase weiterentwickelt

werden kann (Anforderung 13). Eine Übertragung auf unterschiedliche Plattformen ist konzeptionell zwar möglich, jedoch fand in der Implementierung die Verwendung von plattform-gebundenen Werkzeugen statt. Dadurch ist die Wunschforderung 14 nur teilweise erfüllt. Die letzte Anforderung an den Digitalen Zwilling unter der Nummer 14 konnte ebenfalls nur teilweise erfüllt werden. Wie im vorigen Kapitel zur Bewertung der Validierung diskutiert, stößt die Ausführungsgeschwindigkeit des Digitalen Zwillings an Grenzen und kann nicht mit der Geschwindigkeit des physischen Systems mithalten.

Im letzten Abschnitt werden die Anforderungen an die Modelle des Digitalen Zwillings betrachtet. Wie bereits in den Anforderungen 12 und 14 konnte keine vollkommene Neutralität bzw. Unabhängigkeit von proprietären Werkzeugen erreicht werden. Mangels Alternativen mussten im Rahmen der Implementierung spezialisierte Werkzeuge verwendet werden, um insbesondere die Verhaltenslogik zu virtualisieren. Dadurch konnten nicht alle der verwendeten Modelle in einem neutralen Format bereitgestellt werden und die Anforderung 16 ist nicht für alle Partialmodelle des Digitalen Zwillings erfüllt worden. Durch den modularen Aufbau ist es möglich, den Digitalen Zwilling anwendungsbezogen zu konfigurieren, wodurch die Anforderungen 17 und 18 erfüllt wurden. Gleichzeitig wird die Komplexität der einzelnen Module hinter Schnittstellen verborgen (Anforderung 19). Schließlich wurde sowohl methodisch als auch praxisbezogen beschrieben, wie die Virtuelle Klemmleiste zu generieren und zu konfigurieren ist, womit die letzte Anforderung ebenfalls erfüllt wurde.

6.5 Fazit

Das entwickelte Konzept einer Virtuellen Klemmleiste zur Integration der Verhaltenslogik in den Digitalen Zwilling konnte anhand der Implementierung und unter Berücksichtigung zweier Anwendungsfälle am Beispiel des CNC Laserplotters validiert werden. Die Prozessbeschreibung und das Informationsmodell unterstützten dabei das schrittweise und systematische Vorgehen bei der Umsetzung des Konzepts. Mögliche weiterführende Forschung wird im nachfolgenden Kapitel 7 aufgegriffen.

Die im Kapitel 3.4 erarbeiteten Anforderungen an das Konzept konnten innerhalb der betrachteten Grenzen überwiegend erfüllt werden. Die eingeschränkte Verbreitung der OPC UA Technologie führte dazu, dass diese nicht in allen Bereichen des Digitalen Zwillings eingesetzt werden konnte. Weiterhin konnten nicht alle Partialmodelle des Digitalen Zwillings in neutralen Datenformaten und unabhängig von Plattformen und spezialisierten Autorenwerkzeugen umgesetzt werden. Dies führte zu einer Abhängigkeit von Softwarewerkzeugen, die zum aktuellen Zeitpunkt nicht für den durchgehenden Betrieb, welchen ein Digitaler Zwilling erfordert, ausgelegt sind. Zuletzt wurde durch

den notwendigen Umfang der Datenverarbeitung und Kommunikation nicht die gleiche Ausführungsgeschwindigkeit des Digitalen Zwillings erreicht, wie der Physische Zwilling dies vorgibt. Hierfür ist sowohl weitere Forschung als auch weiterer technischer Fortschritt notwendig.

Tabelle 6.1: Verifikation des Anforderungsprofils

Nr.	Art	Bezeichnung	Grad
Anforderungen an die eingebettete Software			
1	F	*Die eingebettete Software des Physischen Zwillings muss in den Digitalen Zwilling integriert werden können.*	●
2	F	*Die eingebettete Software muss ohne physische eingebettete Hardware ausgeführt werden können.*	●
3	W	*Der Digitale Zwilling soll die gleiche Software verwenden, welche auch im Physischen Zwilling eingesetzt wird.*	●
Anforderungen an die virtuelle Software-Hardware Schnittstelle			
4	F	*Es muss eine Schnittstelle zwischen der eingebetteten Software und den davon abhängigen Modulen des Digitalen Zwillings existieren.*	●
5	F	*Das Informationsmodell dient als Grundlage für die Generierung der Schnittstelle im Digitalen Zwilling.*	●
6	F	*Die virtuelle Software-Hardware Schnittstelle muss so konzipiert sein, dass sowohl bei einer Neuentwicklung, als auch beim Aufrüsten eines bestehenden Systems diese verwendet werden kann.*	●
7	F	*Die virtuelle Software-Hardware Schnittstelle muss erweiterbar sein.*	●
8	F	*Die virtuelle Software-Hardware Schnittstelle muss domänenübergreifend eingesetzt werden können.*	◒
9	F	*Bei der Erstellung einer virtuellen Software-Hardware Schnittstelle und des zugrundeliegendes Informationsmodells muss eine methodische und reproduzierbare Vorgehensweise angewendet werden.*	●
10	F	*Die Erstellung und Erweiterung der virtuellen Software-Hardware Schnitt- stelle und des zugrundeliegendes Informationsmodells soll mittels geeigneter Werkzeuge unterstützt und vereinfacht werden.*	●

Tabelle 6.2: Verifikation des Anforderungsprofils (Fortsetzung)

Nr.	Art	Bezeichnung	Grad
Anforderungen an den Digitalen Zwilling			
11	F	*Der Physische und der Digitale Zwilling müssen eine bidirektionale Verbindung besitzen.*	●
12	W	*Der Digitale Zwilling soll ohne domänenspezifische Autorenwerkzeuge einsetzbar sein.*	◓
13	F	*Der Digitale Zwilling muss auch in der Nutzungsphase weiterentwickelt werden können.*	●
14	W	*Die Grundstruktur des Digitalen Zwillings sollte auf unterschiedliche Plattformen übertragen werden können.*	◓
15	W	*Die Ausführungsgeschwindigkeit des Digitalen Zwillings soll möglichst an das Verhalten des realen Systems angepasst sein.*	◓
Anforderungen an die Modelle und die Konfiguration			
16	W	*Die im Digitalen Zwilling eingesetzten Modelle sollen in einem neutralen Datenformat vorliegen.*	◓
17	F	*Der Digitale Zwilling muss anwendungsbezogen konfiguriert werden können.*	●
18	F	*Der Digitale Zwilling muss modular aufgebaut sein.*	●
19	W	*Zur Reduzierung der Komplexität soll die Funktion der einzelnen Module des Digitalen Zwillings hinter Schnittstellen verborgen werden.*	●
20	W	*Der Anwender soll bei der Konfiguration der virtuellen Software-Hardware Schnittstelle unterstützt werden.*	●

Legende: ● = vollständig erfüllt; ◓ = teilweise erfüllt; ○ = nicht erfüllt

7 Ausblick

Während der Erarbeitung des Stands der Forschung, der Entwicklung des Konzepts, der Durchführung der Implementierung und der abschließenden Validierung entstanden weitere wissenschaftliche Fragestellungen, die entweder in die Dissertation mit einflossen oder über den Rahmen der Dissertation hinaus gingen. Im vorliegenden Kapitel wird ein Ausblick auf diese offenen Forschungspotenziale gegeben.

Das entwickelte Konzept der Virtualisierung der Verhaltenslogik kann erweitert werden, indem aus der Disziplin der Elektronik ein Ansatz eingebunden wird, die eingebettete Elektronik teil- oder vollautomatisiert in ein ausführbares Modell, welches im Digitalen Zwilling eingesetzt werden kann, abzuleiten. In diesem Kontext kann das im Rahmen von *AUTOSAR* entwickelte Konzept einer Abstraktionsschicht für Mikrocontroller *AUTOSAR MCAL* (*Micro Controller Abstaction Layer*) [16] analysiert und möglicherweise übertragen werden. Bei diesem Ansatz wird die Schnittstelle zum Mikrocontroller abstrahiert, wodurch im Digitalen Zwilling der Umfang von individuellen Modellen reduziert werden könnte.

Eine weitere Forschungsfragestellung betrifft die Partialmodelle des Digitalen Zwillings. Im Rahmen der Dissertation wurden einige Partialmodelle verwendet und vorgestellt. Die Thematik der Ableitung hersteller- bzw. plattform-unabhängiger Partialmodelle aus dem Digitalen Master ist wissenschaftlich noch nicht vollständig durchdrungen. Ein mögliches Rahmenwerk für die Verknüpfung und die Konfiguration von Partialmodellen eines Digitalen Zwillings auf einer neutralen Plattform stellt der web-basierte Digitale Zwilling dar (vgl.: [85, 78, 124]. Die ersten Ansätze müssen weiterentwickelt werden, um den vollen Umfang eines Digitalen Zwillings web-basiert nutzen zu können.

Bei einigen industriellen Anwendungsfällen muss im Rahmen der Echtzeitfähigkeit eine definierte Geschwindigkeit und Vorhersagbarkeit gegeben sein. Mit steigendem Umfang des Digitalen Zwillings und den konventionellen Methoden nimmt die Geschwindigkeit jedoch tendenziell ab, während die Fehleranfälligkeit zunimmt. Um diesem Trend entgegenzuwirken, müssen insbesondere im Kontext der Simulation neue Wege gefunden werden. Im Gegensatz zu den konventionellen Simulationsverfahren, welche aus der Produktentwicklung stammen, müssen die Simulationen in einem Digitalen Zwilling anderen Anforderungen gerecht werden. Dazu gehört insbesondere die Möglichkeit, unterschiedliche Simulationsarten zu kombinieren und eine Simulationskette aus mehreren

Simulationen kontinuierlich auszuführen, statt iterativ einzelne Inputdecks zu verarbeiten. Auf diesem Gebiet besteht ebenfalls Forschungspotenzial.

Weiterhin besteht Bedarf in der Erforschung der bidirektionalen Verbindung zwischen dem Digitalen und Physischen Zwilling. Grundsätzlich kann eine Verbindung hergestellt werden. Ein möglicher Ansatz wurde im Rahmen der Dissertation beschrieben. Es ist jedoch wissenschaftlich noch nicht durchdrungen, wie die gegenseitige Beeinflussung, die als Zirkularität bezeichnet wurde, ausgeprägt und reguliert werden kann, damit keine unerwünschten Auswirkungen auf die Systeme entstehen. Die Beherrschung dieses Sachverhalts wird in zukünftigen Anwendungen adressiert werden müssen.

Schließlich wurden im Rahmen des Konzepts Annahmen getroffen, die zum aktuellen Stand nicht immer erfüllt sind. Dies betrifft beispielsweise die erforderlichen Informationen aus dem Quellcode des Systems, welche im Falle kommerzieller Software nicht bekannt sind. Weiterhin werden Modelle aus der Produktentwicklung herangezogen, die ebenfalls den Schutzrechten des Herstellers unterliegen. Hierfür müssen Konzepte und rechtliche Rahmenbedingungen erarbeitet werden, damit der Digitale Zwilling diese Modelle und Informationen aus der Produktentwicklung nutzen kann.

Diese Forschungsfelder erwiesen sich im Kontext der Dissertation als bedeutsam und müssen weiter untersucht werden, um das Zukunftsbild des Digitalen Zwillings zu verwirklichen.

8 Zusammenfassung

Die Vision eines Digitalen Zwillings beschreibt ein durchgängiges Abbild eines Systems, welches in diversen Anwendungsfällen als virtuelle Repräsentation eingesetzt werden kann und in einer bidirektionalen Verbindung mit dem Physischen Zwilling steht. Der Digitale Zwilling soll dabei Eigenschaften des physischen Systems abbilden können, zu den auch das Verhalten gehört.

Um das Verhalten von mechatronischen Systemen abbilden zu können, muss auch die Verhaltenslogik, zu der insbesondere die eingebettete Software und die Elektronik gehören, integriert werden. Zum aktuellen Stand der Technik existieren unterschiedliche Ansätze, das Verhalten zu integrieren. Diese verwenden dazu jedoch unterschiedliche Ersatzmodelle, um das Verhalten nachzubilden. Dies resultiert in möglichen Abweichungen vom tatsächlichen Verhalten und in grundsätzlichen Unterschieden in der Verhaltenslogik. Daraus resultiert der Forschungsbedarf nach einer Möglichkeit der Abbildung des Verhaltens auf Basis der tatsächlichen Verhaltenslogik. Eine originalgetreue Abbildung des Verhaltens stellt die Grundlage für neuartige Anwendungsfälle, wie z.B. die Analyse und Vorhersage des dynamischen Verhaltens eines Systems infolge geänderter Rahmenbedingungen.

Aus dieser Motivation und dem Stand der Technik leitet sich die Zielsetzung der Dissertation ab. Das Ziel der Dissertation ist die Entwicklung einer Virtuellen Klemmleiste zur Integration der Verhaltenslogik in den Digitalen Zwilling. Im Fokus liegt dabei die Nutzung der originalen eingebetteten Software, die mittels einer virtuellen Abbildung der Elektronik befähigt wird, auf der Plattform des Digitalen Zwillings eingesetzt und mittels einer *Virtuellen Klemmleiste* an dessen Partialmodelle angebunden zu werden.

Zur Erreichung des Ziels wurde ein Konzept entwickelt, welches die Beschreibung des Prozesses der Integration des Verhaltens auf Basis interdisziplinärer Zusammenarbeit und die Entwicklung einer *Virtuellen Klemmleiste* als Schnittstelle der virtuellen Verhaltenslogik beinhaltet. Die Integration der Verhaltenslogik in den Digitalen Zwilling besteht demnach aus den zwei Teilprozessen der *Virtualisierung der Verhaltenslogik* und der *Erweiterung eines Digitalen Zwillings durch virtualisierte Verhaltenslogik*. Zur Durchführung der Integration wird die physische Verhaltenslogik, die individuellen Anforderungen und die Partialmodelle als Eingabe benötigt. Als Ressourcen und Werkzeuge werden unterschiedliche Modellierungs- und Simulationswerkzeuge sowie der Physische Zwilling und

geeignete Kommunikationstechnologien herangezogen. Gesteuert und durchgeführt wird der Prozess durch den Einsatz von Experten aus den Gebieten der Softwaretechnik, der Elektronik, der Simulationstechnik sowie einem Systemexperten, welcher die übergeordnete Schnittstelle zwischen den Domänen bildet. Zusätzlich wird im Konzept eine informationsmodell-basierte Schnittstelle erarbeitet, welche als generische Verbindung zwischen der virtuellen Verhaltenslogik und den anschließenden Partialmodellen des Digitalen Zwillings eingesetzt wird. Die Ausgabe des Gesamtprozesses ist schließlich ein Digitaler Zwilling mit integrierter Verhaltenslogik.

Zur Validierung der Tragfähigkeit des Konzepts ist der Prozess am Beispiel eines CNC Laserplotters durchgeführt und implementiert worden. Bei der Implementierung wurden sowohl generische Werkzeuge eingesetzt, als auch individuelle Anwendungen entwickelt, um die unterschiedlichen Aufgaben bei der Integration der Verhaltenslogik abzudecken.

Die Implementierung wurde anschließend genutzt, um eine Validierung und Verifikation des Anforderungsprofils auf Basis von zwei konkreten Anwendungsfällen durchzuführen. Neben dem inhärenten Anwendungsfall der *Integration der Verhaltenslogik in den Digitalen Zwilling* wurde die *Steuerung des Digitalen Zwillings mittels der integrierten Verhaltenslogik* beschrieben und validiert. Die Validierung hat gezeigt, dass das entwickelte Konzept geeignet ist, um das originalgetreue Verhalten in den Digitalen Zwilling zu integrieren. Die Verifikation belegte, dass alle Anforderungen mindestens teil-erfüllt sind. Die Validierung und Verifikation bestätigten somit die Tragfähigkeit der Konzeptionierung.

Die vorliegende Dissertation führt zu weiteren Forschungsfragen auf dem adressierten Wissensgebiet. In einem Ausblick werden die bestehenden Lücken und weiterer Forschungsbedarf dargelegt.

9 Anhang

9.1 Anhang A - Implementierung OPC UA Client

```
111     def remoteControl(self, client, *args):
112         if(client==None):
113             print("No OPCUA Server connected")
114             print("RemoteControl is not available")
115         else:
116             print("You can end RemoteControl via Ctrl+C\n")
117             while True:
118                 try:
119                     # get nodes from OPC UA Server
120                     node_posX =  client.get_node("ns=4;i=6116") #Laserplotter.RemoteControl.PositionX
121                     node_posY = client.get_node("ns=4;i=6125") #Laserplotter.RemoteControl.PositionY
122                     node_posZ = client.get_node("ns=4;i=6126") #Laserplotter.RemoteControl.PositionZ
123                     node_spX = client.get_node("ns=4;i=6127") #Laserplotter.RemoteControl.SpeedX
124                     node_spY = client.get_node("ns=4;i=6128") #Laserplotter.RemoteControl.SpeedY
125                     node_spZ = client.get_node("ns=4;i=6129") #Laserplotter.RemoteControl.SpeedZ
126
127                     # get values from OPC UA nodes
128                     nodes = [node_posX, node_posY, node_posZ, node_spX, node_spY, node_spZ]
129                     result = client.get_values(nodes)
130
131                     # edit values
132                     positionX = str(round(result[0],2)*-1)
133                     positionY = str(round(result[1],2)*-1)
134                     positionZ = str(round(result[2],2))
135                     speedX = str(round(abs(result[3]),2))
136                     speedY = str(round(abs(result[4]),2))
137                     speedZ = str(round(abs(result[5]),2))
138
139                     # aggregate feed from speed of the different axes
140                     # convert mm/s in mm/min
141                     spX = float(speedX)*60
142                     spY = float(speedY)*60
143                     spZ = float(speedZ)*60
144
145                     # for diagonal feed
146                     if((spX > 100) and (spY > 100)):
147                         speed = round((spX**2+spY**2)**0.5,2)
148
149                     # feed if only one axis is active
150                     # prevent to low feedrates
```

Abbildung 9.1: Implementierung der Funktion *remoteControl* zur Anbindung von *bCNC* an einen OPC UA Server in *control.py* (Fortsetzung auf der nächsten Seite)

```
151                     else:
152                         if ((spX > spY) and (spX > spZ)):
153                             if (spX < 300):
154                                 speed = 1200
155                             else:
156                                 speed = spX
157                         elif((spY > spX) and (spY > spZ)):
158                             if (spY < 300):
159                                 speed = 1200
160                             else:
161                                 speed = spY
162                         elif((spZ > spX) and (spZ > spY)):
163                             spZ = float(speedZ)*60
164                             if (spZ < 200):
165                                 speed = 600
166                             else:
167                                 speed = spZ
168                         else:
169                             speed = 1200
170
171                         # create and send G-Code
172                         gcode = "G1 X" + positionX + " Y" + positionY + " Z" + positionZ + " F"+str(speed)
173                         print(gcode)
174                         self.sendGCode(gcode)
175                         time.sleep(0.1)
176                     except KeyboardInterrupt:
177                         break
```

```
522     def quitOPCUA(self, client):
523         client.disconnect()
524         print("Client disconnected")
525         self.quit()
```

Abbildung 9.2: Fortsetzung der Implementierung der Funktion *remoteControl* und die Implementierung der Funktion zur Trennung der Verbindung *quitOPCUA* in *__main__.py* von *bCNC*

Literatur

[1] Eberhard Abele, Reiner Anderl, Joachim Metternich, Alexander Arndt, Andreas Wank, Oleg Anhokin, Tobias Meudt und Markus Sauer. *Effiziente Fabrik 4.0: Industrie 4.0 - Potentiale, Nutzen und Good-Practice-Beispiele für die hessische Industrie: Zwischenbericht zum Projekt Effiziente Fabrik 4.0*. Bamberg: Meisenbach Verlag, 2015. ISBN: ISBN 978-3-87525-389-4. DOI: 10.13140/RG.2.1.1177.5448.

[2] Acatech. „Cyber-Physical Systems: Innovationsmotor für Mobilität, Gesundheit, Energie und Produktion“. In: *acatech POSITION* (2011).

[3] Elmar Ahle, Christian Bertsch, Jonathan Neudorfer, Siva Sankar Arumugham, Karthikeyan Ramachandran und Andreas Thuy. „FMI for Physical Models on Automotive Embedded Targets“. In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2015, S. 43–50. DOI: 10.3384/ecp1511843.

[4] Kazi Masudul Alam und Abdulmotaleb El Saddik. „C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems“. In: *IEEE Access* 5 (2017), S. 2050–2062. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2657006.

[5] Reiner Anderl. „Industrie 4.0 – Technological approaches, use cases, and implementation“. In: *at - Automatisierungstechnik* 63.10 (2015). ISSN: 0178-2312. DOI: 10.1515/auto-2015-0025.

[6] Reiner Anderl. „Informationstechnologie: Dubbel - Taschenbuch für den Maschinenbau“. In: *Dubbel - Taschenbuch für den Maschinenbau*. Hrsg. von Beate Bender und Dietmar Göhlich. Bd. 2. 2020, S. 89–116.

[7] Reiner Anderl. „Integriertes Produktmodell“. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 84.11 (1989), S. 640–644. ISSN: 0947-0085. DOI: 10.1515/zwf-1989-841124.

[8] Reiner Anderl. „Virtuelle Produktentwicklung: Dubbel - Taschenbuch für den Maschinenbau“. In: *Dubbel - Taschenbuch für den Maschinenbau*. Hrsg. von Beate Bender und Dietmar Göhlich. Bd. 2. 2020, S. 117–140.

[9] Reiner Anderl, Sebastian Haag, Klaus Schützer und Eduardo Zancul. „Digital twin technology – An approach for Industrie 4.0 vertical and horizontal lifecycle integration“. In: *it - Information Technology* 60.3 (2018), S. 125–132. ISSN: 1611-2776. DOI: 10.1515/itit-2017-0038.

[10] Reiner Anderl, Roland Nattermann und Thomas Rollmann. „Das W-Modell - Systems Engineering in der Entwicklung aktiver Systeme“. In: *PLM-Portal* (2012).

[11] Reiner Anderl, Daniel Strang, André Picard und Alexander Christ. „Integriertes Bauteildatenmodell für Industrie 4.0“. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 109.1-2 (2014), S. 64–69. ISSN: 0947-0085. DOI: 10.3139/104.111098.

[12] Reiner Anderl und Dietmar Trippner. *STEP, Standard for the Exchange of Product Model Data: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*. Stuttgart und Leipzig: Teubner, 2000. ISBN: 978-3-519-06377-3. DOI: 10.1007/978-3-322-89096-2.

[13] Zeeshan Anwar. *Reusable Device Simulation Models for Embedded System Virtual Platforms*. Hrsg. von Design and Reuse. 2021.

[14] Arm Ltd. *Fixed Virtual Platforms – Pre-configured Simulation Models – ARM*. 24.05.2022. URL: https://www.arm.com/products/development-tools/simulation/fixed-virtual-platforms (besucht am 24. 05. 2022).

[15] Behrang Ashtari Talkhestani, Tobias Jung, Benjamin Lindemann, Nada Sahlab, Nasser Jazdi, Wolfgang Schloegl und Michael Weyrich. „An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System“. In: *at - Automatisierungstechnik* 67.9 (2019), S. 762–782. ISSN: 0178-2312. DOI: 10.1515/auto-2019-0039.

[16] AUTOSAR. *Classic Plattform - Specification of MCU Driver*. 2017-12-08. URL: https://www.autosar.org/ (besucht am 20. 09. 2022).

[17] Jens Bastian, Christoph Clauß, Susann Wolf und Peter Schneider. „Master for Co-Simulation Using FMI“. In: *Proceedings from the 8th International Modelica Conference, Technical Univeristy, Dresden, Germany*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2011, S. 115–120. DOI: 10.3384/ecp11063115.

[18] Thomas Bauernhansl. „Industrie 4.0: Entwicklungsfelder für den Mittelstand: Aktuelle Hemmnisse und konkrete Bedarfe“. In: (2016).

[19] Beate Bender, Kilian Gericke, Gerhard Pahl und Wolfgang Beitz, Hrsg. *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung*. 9. Auflage. Springer eBook Collection. Berlin und Heidelberg: Springer Vieweg, 2021. ISBN: 9783662573037. DOI: 10.1007/978-3-662-57303-7.

[20] Beate Bender und Dietmar Göhlich, Hrsg. *Dubbel - Taschenbuch für den Maschinenbau*. 2020.

[21] Alan Berg. *Jenkins Continuous Integration Cookbook*. 1st ed. Olton: Packt Publishing Ltd, 2012. ISBN: 9781849517416.

[22] Christoph Berger, Andreas Hees, Stefan Braunreuther und Gunther Reinhart. „Characterization of Cyber-Physical Sensor Systems“. In: *Procedia CIRP* 41 (2016), S. 638–643. ISSN: 22128271. DOI: 10.1016/j.procir.2015.12.019.

[23] Christian Bertsch, Elmar Ahle und Ulrich Schulmeister. „The Functional Mockup Interface - seen from an industrial perspective“. In: *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2014, S. 27–33. DOI: 10.3384/ecp1409627.

[24] Bibliographisches Institut. *Duden online Wörterbuch*. 2021.

[25] BITKOM e.V., VDMA e.V. und ZVEI e.V., Hrsg. *Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0*. 2015.

[26] Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson und Antoine Viel. „Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models“. In: *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2012, S. 173–184. DOI: 10.3384/ecp12076173.

[27] Torsten. Blochwitz, Martin. Otter, Martin. Arnold, Constanze. Bausch, Christoph. Clauss, Hilding. Elmqvist, Andreas. Junghanns, Jakob. Mauss, Manuel. Monteiro, Thomas. Neidhold, Dietmar. Neumerkel, Hans. Olsson, Jörg.-Volker. Peetz und Susann. Wolf. „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“. In: *Proceedings from the 8th International Modelica Conference, Technical Univeristy, Dresden, Germany*. Linköping Electronic Confe-

rence Proceedings. Linköping University Electronic Press, 2011, S. 105–114. DOI: 10.3384/ecp11063105.

[28] BMWi, Hrsg. *Details Of the Administration Shell: Part 1 - The exchange of information between partners in the value chain of Industrie 4.0*. 2020.

[29] BMWi, Hrsg. *Details of the Asset Administration Shell: Part 2 - Interoperability at Runtime – Exchanging Information via Application Programming Interfaces*. 2021.

[30] BMWi, Hrsg. *Struktur der Verwaltungsschale: Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente*. 2016.

[31] Stefan Boschert und Roland Rosen. „Digital Twin—The Simulation Aspect“. In: *Mechatronic Futures*. Hrsg. von Peter Hehenberger und David Bradley. Cham: Springer International Publishing, 2016, S. 59–74. ISBN: 978-3-319-32156-1.

[32] David Broman, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis und Michael Wetter. „Requirements for hybrid cosimulation standards“. In: *Proceedings of the 18th International Conference on Hybrid Systems Computation and Control*. Hrsg. von Antoine Girard. New York, NY: ACM, 2015, S. 179–188. ISBN: 9781450334334. DOI: 10.1145/2728606.2728629.

[33] Manfred Broy, Hrsg. *Cyber-Physical Systems: Innovation durch Software-Intensive eingebettete Systeme*. Springer, 2010.

[34] Manfred Broy. „Engineering Cyber-Physical Systems: Challenges and Foundations“. In: *Complex Systems Design & Management*. Hrsg. von Marc Aiguier, Yves Caseau, Daniel Krob und Antoine Rauzy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 1–13. ISBN: 978-3-642-34404-6.

[35] Manfred Broy, María Victoria Cengarle und Eva Geisberger. „Cyber-Physical Systems: Imminent Challenges“. In: *Large-scale complex IT systems*. Hrsg. von Radu Calinescu und David Garlan. Bd. 7539. Lecture Notes in Computer Science. Berlin: Springer, 2012, S. 1–28. ISBN: 978-3-642-34058-1. DOI: 10.1007/978-3-642-34059-8_1.

[36] Luca P. Carloni, Roberto Passerone, Alessandro Pinto und Alberto L. Angiovanni-Vincentelli. „Languages and Tools for Hybrid Systems Design“. In: *Foundations and Trends in Electronic Design Automation* 1.1/2 (2006), S. 1–193. ISSN: 1551-3939. DOI: 10.1561/1000000001.

[37] Chamnit. *Grbl GitHub Webpage*. 2021. URL: https://github.com/gnea/grbl-Mega (besucht am 24. 08. 2022).

[38] Sang-Young Cho und Jeong-Bae Lee. „Virtual Development Environment for Embedded Systems Using ARMulator and SystemC Models“. In: *Intelligent technical systems*. Hrsg. von Natividad Martínez Madrid und Ralf E. Seepold. Lecture Notes in Electrical Engineering. Berlin: Springer, 2009, S. 59–72. ISBN: 978-1-4020-9823-9. DOI: 10.1007/978-1-4020-9823-9_5.

[39] Horst Czichos. *Mechatronik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. ISBN: 978-3-658-26293-8. DOI: 10.1007/978-3-658-26294-5.

[40] Giovanni Dal Maso, Diego Rovere, Paolo Pedrazzoli, Marino Alge und Michele Ciavotta. „A Centralized Support Infrastructure (CSI) to Manage CPS Digital Twin, towards the Synchronization between CPSs Deployed on the Shopfloor and Their Digital Representation“. In: 2019, S. 317. ISBN: 9788770220415.

[41] Violeta Damjanovic-Behrendt und Wernher Behrendt. „An open source approach to the design and implementation of Digital Twins for Smart Manufacturing“. In: *International Journal of Computer Integrated Manufacturing* 32.4-5 (2019), S. 366–384. ISSN: 0951-192X. DOI: 10.1080/0951192X.2019.1599436.

[42] Thomas Dasbach. *Automatisierte Bestimmung der Demontagereihenfolge für Cyber-Physische Produkte*. 1. Auflage. Bd. 72. Forschungsberichte aus dem Fachgebiet Datenverarbeitung in der Konstruktion. Düren: Shaker, 2022. ISBN: 9783844086867.

[43] Deutsches Institut für Normung. *Qualitätsmanagementsysteme: Grundlagen und Begriffe*. November 2015.

[44] DIN. „DIN 62541-1 OPC UA: Teil 1: Übersicht und Konzepte“. In: (2011).

[45] DIN. „DIN SPEC 91345 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)“. In: (2016).

[46] Luiz Fernando C. S. Durão, Sebastian Haag, Reiner Anderl, Klaus Schützer und Eduardo Zancul. „Digital Twin Requirements in the Context of Industry 4.0“. In: *Proceedings of the 15th IFIP International Conference on Product Lifecycle Management* (2018), S. 204–214.

[47] EMPHYSIS. *Functional Mock-up Interface for Embedded Systems (eFMI): Standard*. 2021-01-27.

[48] Svenja Falk, Lin An, Reiner Anderl, Henrik Beermann, Zihe Gao, Junhai Li, Steffen Preissler und Kristin Shi-Kupfer. „Wertschöpfungsnetzwerke als Grundlage für digitale Geschäftsmodelle: Anwendungsfälle aus Deutschland und China: Deutsche Kurzfassung“. In: (2021).

[49] Markus Friedrich. „Parallel Co-Simulation for Mechatronic Systems: Dissertration“. Diss. 2011.

[50] Richard. M. Fujimoto. „Parallel and distributed simulation systems“. In: *Proceedings of the 2001 Winter Simulation Conference*. Hrsg. von Brett A. Peters. New York, NY: Assoc. for Computing Machinery, 2001, S. 147–157. ISBN: 0-7803-7307-3. DOI: 10.1109/WSC.2001.977259.

[51] Simon Fürst. „AUTOSAR – A Worldwide Standard is on the Road .“ In: 2009.

[52] Eva Geisberger und Manfred Broy. *Integrierte Forschungsagenda Cyber-Physical Systems: acatech Studie*. Bd. 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-29098-5. DOI: 10.1007/978-3-642-29099-2.

[53] Ralf Gessler. *Entwicklung Eingebetteter Systeme: Vergleich von Entwicklungsprozessen für FPGA- und Mikroprozessor-Systeme : Entwurf auf Systemebene*. 2., aktualisierte und erweiterte Auflage. Lehrbuch. Wiesbaden und Heidelberg: Springer Vieweg, 2020. ISBN: 978-3-658-30548-2. DOI: 10.1007/978-3-658-30549-9.

[54] Claudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen und Hans Vangheluwe. „Co-simulation: State of the Art“. In: *Systems and Control*. 2017.

[55] Antonio Gonzalez, Scott Mahlke, Shubu Mukherjee, Resit Sendag, Derek Chiou und Joshua J. Yi. „Reliability: Fallacy or Reality?“ In: *IEEE Micro* 27.6 (2007), S. 36–45. ISSN: 0272-1732. DOI: 10.1109/MM.2007.107.

[56] Christian Granrath, Max-Arno Meyer, Jakob Andert, Jens Ewald, Roberto Klink, Christoph Stroh, Thinh Pham, Richard Phillips, Carl Hettig, Leonardo Santaroni, Markus Deppe und Omar Hegazy. „EleMA: A reference simulation model architecture and interface standardfor modeling and testing of electric vehicles“. In: (2020). DOI: 10.18154/RWTH-2020-05682.

[57] Iris Gräßler, Alexander Pöhler und Julian Hentze. „Decoupling of Product and Production Development in Flexible Production Environments“. In: *Procedia CIRP* 60 (2017), S. 548–553. ISSN: 22128271. DOI: 10.1016/j.procir.2017.01.040.

[58] Michael Grieves, Hrsg. *Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management*. Space Coast Press, 2011. ISBN: 0982138008.

[59] Michael Grieves und John Vickers. „Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems“. In: *Transdisciplinary Perspectives on Complex Systems*. Hrsg. von Franz-Josef Kahlen, Shannon Flumerfelt und Anabela Alves. Bd. 89. Cham: Springer International Publishing, 2017, S. 85–113. ISBN: 978-3-319-38754-3. DOI: 10.1007/978-3-319-38756-7_4.

[60] Marco Grimm, Reiner Anderl und Yan Wang. „Conceptual Model Based Approach for Multi-Disciplinary Cyber Physical Systems Design and Engineering“. In: *Proceedings of TMCE 2014*. 2014.

[61] Sebastian Haag und Reiner Anderl. „Automated Generation of as-manufactured geometric Representations for Digital Twins using STEP“. In: (2019).

[62] Philipp Hedrich, Nicolas Brötz und Peter F. Pelz. „Resilient Product Development - A New Approach for Controlling Uncertainty“. In: *Applied Mechanics and Materials* 885 (2018), S. 88–101. DOI: 10.4028/www.scientific.net/AMM.885.88.

[63] Peter Hehenberger, Birgit Vogel-Heuser, David Bradley, Benoit Eynard, Tetsuo Tomiyama und Sofiane Achiche. „Design, Modelling, Simulation and Integration of Cyber Physical Systems: Methods and Applications“. In: *Computers in Industry* Oktober 2016.82 (2016). ISSN: 01663615.

[64] Stephan Hensel, Markus Graube, Leon Urbas, Till Heinzerling und Mathias Oppelt. „Co-simulation with OPC UA“. In: *Proceedings, 2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. Piscataway, NJ: IEEE, 2016, S. 20–25. ISBN: 978-1-5090-2870-2. DOI: 10.1109/INDIN.2016.7819127.

[65] Michael Hoffmeister, Andreas Orzelski und Marco Mendes. *AASX - Asset Administration Shell Explorer*. 2022. URL: https://github.com/admin-shell-io.

[66] IEEE. *IEEE SA - IEEE Standard for Distributed Interactive Simulation (DIS) – Communication Services and Profiles*. 2015.

[67] IEEE. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules*. 2010.

[68] Institute of Automation Technology. *Implementierungen der Verwaltungsschale*. 2022. URL: http://www.i40-aas.de/.

[69] ISO. *10303: Industrielle Automatisierungssysteme und Integration. Produktdatendarstellung und -austausch: Teil 1: Ueberblick und grundlegende Prinzipien*. 1995.

[70] ISO/IEC/IEEE. *12207: Systems and software engineering: Software lifecycle processes*. 2019.

[71] David Jones, Chris Snider, Aydin Nassehi, Jason Yon und Ben Hicks. „Characterising the Digital Twin: A systematic literature review“. In: *CIRP Journal of Manufacturing Science and Technology* (2020). ISSN: 17555817. DOI: 10.1016/j.cirpj.2020.02.002.

[72] José Ríos, Juan Carlos Hernández, Manuel Oliva und Fernando Mas. „Product Avatar as Digital Counterpart of a Physical Individual Product: Literature Review and Implications in an Aircraft“. In: 2015. DOI: 10.3233/978-1-61499-544-9-657.

[73] Tobias Jung, Nasser Jazdi und Michael Weyrich. „A survey on dynamic simulation of automation systems and components in the Internet of Things“. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation*. Piscataway, NJ: IEEE, 2017, S. 1–4. ISBN: 978-1-5090-6505-9. DOI: 10.1109/ETFA.2017.8247770.

[74] Tobias Jung, Payal Shah und Michael Weyrich. „Dynamic Co-Simulation of Internet-of-Things-Components using a Multi-Agent-System“. In: *Procedia CIRP* 72 (2018), S. 874–879. ISSN: 22128271. DOI: 10.1016/j.procir.2018.03.084.

[75] Henning Kagermann, Wolfgang Wahlster und Johannes Helbig. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Final report of the Industrie 4.0 Working Group*. 2013.

[76] Alexander Kern und Reiner Anderl. „Using Digital Twin Data for the Attribute-Based Usage Control of Value-Added Networks“. In: *2020 Seventh International Conference on Software Defined Systems (SDS)*. Hrsg. von Mohammad Alsmirat. Piscataway, NJ: IEEE, 2020, S. 29–36. ISBN: 978-1-7281-7219-4. DOI: 10.1109/SDS49854.2020.9143921.

[77] Dimitris Kiritsis. „Closed-loop PLM for intelligent products in the era of the Internet of things“. In: *Computer-Aided Design* 43.5 (2011), S. 479–501. ISSN: 00104485. DOI: 10.1016/j.cad.2010.03.002.

[78] Sergey Konstantinov, Fadi Assad, Wajid Azam, Daniel Vera, Bilal Ahmad und Robert Harrison. „Developing web-based digital twin for industrial cyber-physical systems“. In: (2021).

[79] Martin Krammer, Martin Benedikt, Torsten Blochwitz, Khaled Alekeish, Nicolas Amringer, Christian Kater, Stefan Materne, Roberto Ruvalcaba, Klaus Schuch, Josef Zehetner, Micha Damm-Norwig, Viktor Schreiber, Natarajan Nagarajan, Isidro Corral, Tommy Sparber, Serge Klein und Jakob Lukas Andert. „The Distributed Co-Simulation Protocol for the Integration of Real-Time Systems and Simulation Environments“. In: *Proceedings of the 50th Computer Simulation Conference*. Society for Modeling and Simulation International (SCS), 2018. ISBN: 9781510860261. DOI: 10.22360/SummerSim.2018.SCSC.001.

[80] Frank-Lothar Krause, Hrsg. *Innovationspotenziale in der Produktentwicklung*. 1. Aufl. München: Hanser, 2007. ISBN: 3446406670.

[81] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes und Wilfried Sihn. „Digital Twin in manufacturing: A categorical literature review and classification“. In: (2018). DOI: 10.1016/j.ifacol.2018.08.474.

[82] Ralf Kübler und Werner Schiehlen. „Two Methods of Simulator Coupling“. In: *Mathematical and Computer Modelling of Dynamical Systems* 6.2 (2000), S. 93–113. ISSN: 1387-3954. DOI: 10.1076/1387-3954(200006)6:2;1-M;FT093.

[83] Vladimir Kutscher, Thiago Weber Martins, Johannes Olbort und Reiner Anderl. „Concept for Interaction of Hardware Simulation and Embedded Software in a Digital Twin Based Test Environment“. In: *Procedia CIRP* 104 (2021), S. 999–1004. ISSN: 22128271. DOI: 10.1016/j.procir.2021.11.168.

[84] Vladimir Kutscher, Johannes Olbort, Oleg Anokhin, Lukas Bambach und Reiner Anderl. „Upgrading of legacy systems to cyber-physical systems“. In: *Proceedings of TMCE 2020* (2020).

[85] Vladimir Kutscher, Johannes Olbort, Benjamin Röhm, Christian Plesker und Reiner Anderl. „Web-Based Digital Twin“. In: *IIC Journal of Innovation* (2021), S. 1–18.

[86] Vladimir Kutscher, Johannes Olbort, Carsten Steinhauer und Reiner Anderl. „Model-Based Interconnection of Digital and Physical Twins Using OPC UA“. In: *Advances in Manufacturing, Production Management and Process Control*. Hrsg. von Mrugalska und Di Cecco. Bd. 1216. Advances in Intelligent Systems and Computing. [S.l.]: Springer International Publishing, 2020, S. 178–185. ISBN: 978-3-030-51980-3. DOI: 10.1007/978-3-030-51981-0_23.

[87] Labcenter Electronics Ltd. *PCB Design and Circuit Simulator Software - Proteus*. 2022. URL: https://www.labcenter.com/ (besucht am 24. 05. 2022).

[88] Jürgen Lange, Frank Iwanitz und Thomas J. Burke. *OPC: Von Data Access bis Unified Architecture*. 5., durchgesehene Auflage. Berlin und Offenbach: VDE Verlag GmbH, 2014. ISBN: 9783800735068.

[89] Edward A. Lee. „CPS Foundations“. In: *Proceedings of the 47th Design Automation Conference*. 2010, S. 737–742.

[90] Edward A. Lee. „Cyber Physical Systems: Design Challenges“. In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, S. 363–369. ISBN: 978-0-7695-3132-8. DOI: 10.1109/ISORC.2008.25.

[91] Edward A. Lee. „The Past, Present and Future of Cyber-Physical Systems: A Focus on Models“. In: *Sensors* 15.3 (2015), S. 4837–4869. DOI: 10.3390/s150304837.

[92] Jay Lee, Chao Jin und Zongchang Liu. „Predictive Big Data Analytics and Cyber Physical Systems for TES Systems“. In: *Advances in Through-life Engineering Services*. Hrsg. von Louis Redding, Rajkumar Roy und Andy Shaw. Decision Engineering Ser. Cham: Springer International Publishing, 2017, S. 97–112. ISBN: 978-3-319-49937-6. DOI: 10.1007/978-3-319-49938-37.

[93] Chao Liu, Xiaoyang Hong, Zehuan Zhu und Xun Xu. „Machine Tool Digital Twin: Modelling Methodology and Applications“. In: (2018).

[94] Weichao Luo, Tianliang Hu und Wendan Zhu. „Digital twin modeling method for CNC machine tool: ICNSC 2018 - The 15th IEEE International Conference on Networking, Sensing and Control : March 27-29, 2018, Zhuhai, China“. In: (2018).

[95] Ander Madariaga, Jaime Jiménez, José Luis Martín, Unai Bidarte und Aitzol Zuloaga. „Review of electronic design automation tools for high-level synthesis“. In: *2010 International Conference on Applied Electronics*. 2010, S. 1–6.

[96] Wolfgang Mahnke, Stefan-Helmut Leitner und Matthias Damm. *OPC Unified Architecture*. 1. Aufl. Berlin Heidelberg: Springer Vieweg, 2009. ISBN: 978-3-540-68898-3. DOI: 10.3403/BSEN62541.

[97] Nadja Marko, Jonas Ruebsam, Andreas Biehn und Hannes Schneider. „Scenario-based Testing of ADAS - Integration of the Open Simulation Interface into Co-simulation for Function Validation“. In: *SIMULTECH*. 2019.

[98] Mathworks. *MATLAB - Plattform für Programmierung und numerische Berechnungen*. URL: https://de.mathworks.com/ (besucht am 06. 05. 2022).

[99] Jeff McAffer, Paul VanderLei und Simon Archer. *OSGi and Equinox: Creating highly modular Java systems*. The eclipse series. Upper Saddle River, NJ: Addison-Wesley, 2010. ISBN: 0321585712.

[100] Microchip Technology Inc. *Microchip Studio Webpage*. URL: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio (besucht am 24. 08. 2022).

[101] Elizabeth Minton und Lynn Kahle. *Belief Systems, Religion, and Behavioral Economics: Marketing in Multicultural Environments*. Jan. 2014.

[102] Markus Mirz, Steffen Vogel, Bettina Schafer und Antonello Monti. „Distributed real-time co-simulation as a service“. In: *2018 IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*. IEEE, 31.01.2018 - 02.02.2018, S. 534–539. ISBN: 978-1-5090-4974-5. DOI: 10.1109/IESES.2018.8349934.

[103] Modelica Association. *Distributed Co-Simulation Protocol (DCP): Modelica Association Project*. 2019. URL: https://dcp-standard.org/ (besucht am 12. 05. 2022).

[104] Modelica Association. *Modelica - A Unified Object-Oriented Language for Systems Modeling: Language Specification*. 2017-04-10.

[105] Modelica Association. *Modelica - Simulation Environment*. 2022. URL: https://modelica.org/ (besucht am 06. 05. 2022).

[106] Modelica Association. *System Structure and Parametrization*. 1.04.2022. URL: https://ssp-standard.org/ (besucht am 12. 05. 2022).

[107] Modelica Association Project FMI. „Functional Mock-up Interface for Model Exchange and Co-Simulation: Version 2.0.1“. In: (2019).

[108] László Monostori. „Cyber-physical Production Systems: Roots, Expectations and R&D Challenges“. In: *Procedia CIRP* 17 (2014), S. 9–13. ISSN: 22128271. DOI: 10.1016/j.procir.2014.03.115.

[109] László Monostori, Botond Kádár, Thomas Bauernhansl, Shinsuke Kondoh, Soundar Kumara, Gunther Reinhart, Olaf Sauer, Günther Schuh, Wilfried Sihn und Kanji Ueda. „Cyber-physical systems in manufacturing“. In: *CIRP Annals* 65.2 (2016), S. 621–641. ISSN: 00078506. DOI: 10.1016/j.cirp.2016.06.005.

[110] Pieter J. Mosterman. „An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages“. In: *Hybrid systems*. Hrsg. von Frits W. Vaandrager. Bd. 1569. Lecture Notes in Computer Science. Berlin und Heidelberg: Springer, 1999, S. 165–177. ISBN: 978-3-540-65734-7. DOI: 10.1007/3-540-48983-5_17.

[111] Daniel Mueller-Gritschneder und Andreas Gerstlauer. „Host-Compiled Simulation“. In: *Handbook of Hardware/Software Codesign*. Hrsg. von Soonhoi Ha und Jürgen Teich. Dordrecht: Springer Netherlands, 2017, S. 1–27. ISBN: 978-94-017-7358-4. DOI: 10.1007/978-94-017-7358-4_18-1.

[112] Joerg Neidig, Andreas Orzelski und Stefan Pollmeier. *Asset Administration Shell: Reading Guide*. 2022.

[113] Object Management Group. *Common Object Request Broker Architecture (CORBA): Core Specification*. 2004.

[114] Oliver Lenord. *Standardizing eFMI for embedded systems with physical models in the production code software*. 2019. DOI: 10.13140/RG.2.2.23492.78722.

[115] OPC Foundation. „OPC Unified Architecture: Interoperabilität für Industrie 4.0 und das Internet der Dinge“. In: (2016).

[116] Mathias Oppelt, Oliver Drumm, Benjamin Lutz und A. G. Siemens. „Approach for integrated simulation based on plant engineering data“. In: *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)* (2013), S. 1–4.

[117] OSGi Alliance. *OSGi - The Dynamic Module System for Java*. 2018. URL: https://www.osgi.org/.

[118] Rain Ottis und Peeter Lorents. „Cyberspace: Definition and implications“. In: *Proceedings of the 5th European Conference on Information Management and Evaluation* (2011).

[119] OVP World. *Open Virtual Platforms*. 28.09.2021. URL: https://www.ovpworld.org/ (besucht am 28. 09. 2021).

[120] Mi Jeong Park, Dong Kwan Kim, Won-Tae Kim und Seung-Min Park. „Dynamic Software Updates in Cyber-Physical Systems“. In: *2010 International Conference on Information and Communication Technology Convergence (ICTC 2010)*. Piscataway, NJ: IEEE, 2010, S. 425–426. ISBN: 978-1-4244-9806-2. DOI: 10.1109/ICTC.2010.5674807.

[121] Plattform Industrie 4.0. „Deutschlands Zukunft als Produktionsstandort sichern. Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0: Abschlussbericht des Arbeitskreises Industrie 4.0“. In: (2013).

[122] Plattform Industrie 4.0. *Drei Fragen an... Prof. Dr.-Ing. Reiner Anderl: Aktuelle Forschungsbedarfe*. 2020. URL: https://www.plattform-i40.de/IP/Redaktion/DE/Newsletter/2020/Ausgabe23/2020-05-Drei-Fragen-an-Anderl.html (besucht am 21. 02. 2022).

[123] Plattform Industrie 4.0. „Struktur der Verwaltungsschale: Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente“. In: (2016).

[124] Christian Plesker, Vladimir Kutscher, David Bassauer und Reiner Anderl. „Configuration of a Web-Based Digital Twin using a Modular and Flexible Simulation Chain“. In: *Production Management and Process Control*. AHFE International. AHFE International, 2022. DOI: 10.54941/ahfe1001614.

[125] Benjamin Röhm und Reiner Anderl. „Concept of a System Architecture for a Simulation Data Management In the Digital Twin“. In: *ASME 2021 30th Conference on Information Storage and Processing Systems*. American Society of Mechanical Engineers, 2021. ISBN: 978-0-7918-8479-9. DOI: 10.1115/ISPS2021-65300.

[126] Roland Rosen, Georg von Wichert, George Lo und Kurt D. Bettenhausen. „About The Importance of Autonomy and Digital Twins for the Future of Manufacturing“. In: *IFAC-PapersOnLine* 48.3 (2015), S. 567–572. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.06.141.

[127] Chris Rupp und Stefan Queins. *UML2 glasklar: Praxiswissen für die UML-Modellierung*. 4., aktualisierte und erw. Aufl. München: Hanser, 2012. ISBN: 3446431977. DOI: 10.3139/9783446431973.

[128] Benjamin Schleich, Nabil Anwer, Luc Mathieu und Sandro Wartzack. „Shaping the digital twin for design and production engineering“. In: *CIRP Annals* 66.1 (2017), S. 141–144. ISSN: 00078506. DOI: 10.1016/j.cirp.2017.04.040.

[129] Michael Schluse, Marc Priggemeyer, Linus Atorf und Juergen Rossmann. „Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0“. In: *IEEE Transactions on Industrial Informatics* 14.4 (2018), S. 1722–1731. ISSN: 1551-3203. DOI: 10.1109/TII.2018.2804917.

[130] Uwe Schneider und Georg Disterer, Hrsg. *Taschenbuch der Informatik: Mit ... 99 Tabellen*. 7., neu bearb. Aufl. München: Fachbuchverl. Leipzig im Carl Hanser Verl., 2012. ISBN: 9783446426382.

[131] Frank Schöfer. *Nutzung von Geometriemodellen zur Absicherung von Software in mechatronischen Produkten: Zugl.: Darmstadt, Techn. Univ., Diss., 2005*. Bd. 23. Forschungsberichte aus dem Fachgebiet Datenverarbeitung in der Konstruktion. Aachen: Shaker, 2005. ISBN: 3832243666.

[132] Greyce Schroeder, Charles Steinmetz, Carlos Eduardo Pereira, Ivan Muller, Natanael Garcia, Danubia Espindola und Ricardo Rodrigues. „Visualising the digital twin using web services and augmented reality“. In: *Proceedings, 2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. Piscataway, NJ: IEEE, 2016, S. 522–527. ISBN: 978-1-5090-2870-2. DOI: 10.1109/INDIN.2016.7819217.

[133] Greyce N. Schroeder, Charles Steinmetz, Carlos E. Pereira und Danubia B. Espindola. „Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange“. In: *IFAC-PapersOnLine* 49.30 (2016), S. 12–17. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.11.115.

[134] Greyce N. Schroeder, Charles Steinmetz, Ricardo Nagel Rodrigues, Renato Ventura Bayan Henriques, Achim Rettberg und Carlos Eduardo Pereira. „A Methodology for Digital Twin Modeling and Deployment for Industry 4.0“. In: *Proceedings of the IEEE* 109.4 (2021), S. 556–567. ISSN: 0018-9219. DOI: 10.1109/JPROC.2020.3032444.

[135] Günther Schuh, Pia Walendzik, Melanie Luckert, Martin Birkmeier, Anja Weber und Matthias Blum. „Keine Industrie 4.0 ohne den Digitalen Schatten: Wie Unternehmen die notwendige Datenbasis schaffen“. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 111.11 (2016), S. 745–748. ISSN: 0947-0085.

[136] Mike Shafto, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoingne und Lui Wang. „NASA Modeling, Simulation, Information Technology & Processing Roadmap: Technology Area 11“. In: (2010).

[137] Siemens Digital Industries Software. *EDA Software, Hardware & Tools*. 2022. URL: https://eda.sw.siemens.com/en-US/ (besucht am 24. 05. 2022).

[138] Siemens Software. *Siemens Mechatronic Concept Design - Webpage*. 2022.

[139] Christian Siemers und Sebastian Gerstl. „Grundlagen des Embedded Software Engineering“. In: *Elektronikpraxis* (2019).

[140] Ian Sommerville. *Software Engineering*. 10., aktualisierte Auflage. it - informatik. Hallbergmoos: Pearson, 2018. ISBN: 9783863268350.

[141] Vinicius Souza, Robson Cruz, Walmir Silva, Sidney Lins und Vicente Lucena. „A Digital Twin Architecture Based on the Industrial Internet of Things Technologies“. In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2019, S. 1–2. ISBN: 978-1-5386-7910-4. DOI: 10.1109/ICCE.2019.8662081.

[142] Andreas Spillner und Tilo Linz. *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester : Foundation Level nach ISTQB-Standard*. 6., überarbeitete und aktualisierte Auflage. 2019. ISBN: 3864905834.

[143] Rainer Stark, Reiner Anderl, Klaus-Dieter Thoben und Sandro Wartzack. „WiGeP-Positionspapier: „Digitaler Zwilling““. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 115.special (2020), S. 47–50. ISSN: 0947-0085. DOI: 10.3139/104.112311.

[144] Rainer Stark und Thomas Damerau. „Digital Twin“. In: *CIRP Encyclopedia of Production Engineering*. Hrsg. von Sami Chatti und Tullio Tolio. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 1–8. ISBN: 978-3-642-35950-7. DOI: 10.1007/978-3-642-35950-7_16870-1.

[145] Rainer Stark, Simon Kind und Sebastian Neumeyer. „Innovations in digital modelling for next generation manufacturing system design“. In: *CIRP Annals* 66.1 (2017), S. 169–172. ISSN: 00078506. DOI: 10.1016/j.cirp.2017.04.045.

[146] Stephan Werner, Andreas Lauber, Martijn Koedam, Juergen Becker, Eric Sax und Kees Goossens. „Cloud-based Design and Virtual Prototyping Environment for Embedded Systems“. In: *International Journal of Online and Biomedical Engineering (iJOE)* 12.09 (2016), S. 52–60. ISSN: 2626-8493.

[147] SynopSys. *Silver - Virtual ECU*.

[148] Ivana Tomic, Michael J. Breza, Greg Jackson, Laksh Bhatia und Julie A. McCann. „Design and Evaluation of Jamming Resilient Cyber-Physical Systems“. In: *2018 IEEE International Conference on Internet of Things (iThings)*. IEEE, 2018, S. 687–694. ISBN: 978-1-5386-7975-3. DOI: 10.1109/Cybermatics_2018.2018.00138.

[149] Traumflug. *SimulAVR Github Webpage*. 2022. URL: https://github.com/Traumflug/simulavr (besucht am 24. 08. 2022).

[150] Thomas H.-J. Uhlemann, Christian Lehmann und Rolf Steinhilper. „The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0“. In: *Procedia CIRP* 61 (2017), S. 335–340. ISSN: 22128271. DOI: 10.1016/j.procir.2016.11.152.

[151] Unified Autoamtion GmbH. *UA Expert Homepage*. 2022. URL: https://www.unified-automation.com/products/development-tools/uaexpert.html (besucht am 30. 08. 2022).

[152] Unified Autoamtion GmbH. *UA Modeller Homepage*. 2022. URL: https://www.unified-automation.com/products/development-tools/uamodeler.html (besucht am 30. 08. 2022).

[153] Universal Machine Technology Interface Initiative. *UMATI Showcase Specification Homepage*. 2022. URL: https://showcase.umati.org/ (besucht am 30. 08. 2022).

[154] Hendrik Jan van Randen. *Einführung in UML: Analyse und Entwurf von Software*. Springer eBook Collection. Wiesbaden: Springer Vieweg, 2016. ISBN: 9783658144128. DOI: 10.1007/978-3-658-14412-8.

[155] VDI. *VDI 4520 - Produktmanagement: Einführung und Grundlagen*. 2017.

[156] VDI/VDE. „Entwicklung mechatronischer und cyber-physischer Systeme: VDI/VDE-Richtlinie“. In: (11-2021).

[157] VDMA. „Industrie 4.0 Kommunikation mit OPC UA: Leitfaden zur Einführung in den Mittelstand“. In: (2017).

[158] Verein Deutscher Ingenieure. *VDI 2206 - Entwicklungsmethodik für mechatronische Systeme*. 2004.

[159] Mónica M. Villegas, Cristian Orellana und Hernán Astudillo. „A study of over-the-air (OTA) update systems for CPS and IoT operating systems“. In: *Proceedings of the 13th European Conference on Software Architecture - Volume 2*. Hrsg. von Laurence Duchien. ACM Digital Library. New York,NY,United States: Association for Computing Machinery, 2019, S. 269–272. ISBN: 9781450371421. DOI: 10.1145/3344948.3344972.

[160] Vasilis Vlachoudis. *Github Webseite des Softwareprojekts bCNC*. 2022. URL: https://github.com/vlachoudis/bCNC (besucht am 02. 09. 2022).

[161] Yan Wang. „Resilience Quantification for Probabilistic Design of Cyber-Physical System Networks“. In: *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg* 4.3 (2018), S. 21. ISSN: 2332-9017. DOI: 10.1115/1.4039148.

[162] Tim Weilkiens. *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur*. 3., überarbeitete und aktualisierte Auflage. Heidelberg und München: dpunkt.verlag und Ciando, 2014. ISBN: 9783864915437.

[163] Hao Zhang, Qiang Liu, Xin Chen, Ding Zhang und Jiewu Leng. „A Digital Twin-Based Approach for Designing and Multi-Objective Optimization of Hollow Glass Production Line“. In: *IEEE Access* 5 (2017), S. 26901–26911. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2766453.

[164] Wenjun. J. Zhang und C. A. van Luttervelt. „Toward a resilient manufacturing system“. In: *CIRP Annals* 60.1 (2011), S. 469–472. ISSN: 00078506. DOI: 10.1016/j.cirp.2011.03.041.

[165] Pai Zheng und Abinav Shankar Sivabalan. „A generic tri-model-based approach for product-level digital twin development in a smart manufacturing environment“. In: *Robotics and Computer-Integrated Manufacturing* 64 (2020), S. 101958. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2020.101958.