TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Dissertation

Agile Automatisierung von Fertigungszellen

Semiotische Interoperabilität zustandsbehafteter, deterministischer Anlagenkomponenten

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften (Dr. techn.), eingereicht an der TU Wien, Fakultät für Maschinenwesen und Betriebswissenschaften, von

Thomas Fabian Trautner

Mat.Nr.: 01328846

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Burkhard Kittl
Institut für Fertigungstechnik und Photonische Technologien, E311

Wien, Februar 2021

Begutachtet von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn.
Wolfgang Kastner
Institut für Computer Engineering
Treitlstraße 3, 1040 Wien

Univ.Prof. Dipl.-Ing. Dr.techn.
Friedrich Bleicher
Institut für Fertigungstechnik und
Photonische Technologien
Getreidemarkt 9, 1060 Wien

Diese Arbeit wurde von der Österreichischen Forschungsförderungsgesellschaft (FFG) im Rahmen des *Austrian Competence Center for Digital Production* (854187) und der *TU Wien Pilotfabrik Industrie 4.0* (852105) unterstützt.

Ich nehme zur Kenntnis, dass ich zur Drucklegung dieser Arbeit nur mit Bewilligung der Prüfungskommission berechtigt bin.

Eidesstattliche Erklärung:
Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde. Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.
Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.
Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis „Code of Conduct“ an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wien, Februar 2021

Unterschrift

Ich danke meinem Doktorvater Burkhard für die hervorragende wissenschaftliche Betreuung bei der Ausarbeitung und Umsetzung der Dissertation sowie den vielen Personen in meinem Forschungsbereich, die mich durch ihre anregenden Diskussionen inspiriert und vorangetrieben haben. Von diesen möchte ich besonders meinen Freund Iman hervorheben, der mich zu dieser Dissertation motiviert hat.

Mein besonderer Dank gilt außerdem meinen Eltern Renate und Günter, die jederzeit hinter mir gestanden und mir diesen Schritt erst ermöglicht haben. Letztlich gilt mein Dank ebenfalls meiner Partnerin Helena für ihre Unterstützung in all diesen Jahren.

Vorwort

Übersetzungen aus dem Englischen wurden in dieser Arbeit selbst und nach bestem Wissen und Gewissen gemacht. Je nach Art der Zitation wurden die Originalzitate ebenfalls als Fußnote hinterlegt, um eine eindeutige Interpretation zu ermöglichen. Geschlechtsneutrale Bezeichnungen wurden im Rahmen der Arbeit bevorzugt. In Einzelfällen mussten jedoch Begriffe entweder zugunsten der Genauigkeit der Sprache, der besseren Lesbarkeit, aufgrund eines Mangels an gender-gerechten Bezeichnungen in der deutschen Sprache, oder wenn ein ausschließlicher Sachbezug (z. B. bei Herstellerfirma) vorliegt, verwendet werden, die eine geschlechtsspezifische Rollenzuordnung implizieren könnten. Hierbei wird ausdrücklich darauf hingewiesen, dass sich der Inhalt dieser Aussagen auf jedes Geschlecht beziehen soll und keine Benachteiligung eines Geschlechts dadurch impliziert werden soll.

Teilaspekte der in dieser Dissertation bearbeiteten Inhalte wurden von mir bereits in einer ersten Veröffentlichung konzeptuell ausgearbeitet und überprüft:

- **T. Trautner, I. Ayatollahi, D. Strutzenberger u. a., „Behavioral modeling of manufacturing skills in OPC UA for automated execution by an independent cell controller," in *Procedia CIRP*, 2020, In Press.**

Im Rahmen der Themenentwicklung haben des Weiteren folgende, von mir betreute Abschlussarbeiten, zu dieser Dissertation beigetragen:

- B. J. Wallner, *Konzipierung und Umsetzung unterschiedlicher Steuerungsarchitekturen zur agilen Automatisierung von Fertigungszellen*. Wien, 2020. Adresse: https://repositum.tuwien.ac.at/urn:nbn:at:at-ubtuw:1-136183
- F. Lang, *Prozessmodellierung für eine flexible Fertigungszelle*. Wien, 2020. Adresse: https://repositum.tuwien.ac.at/urn:nbn:at:at-ubtuw:1-135484

Kurzfassung

Die europäische Fertigungsindustrie ist von der zunehmenden Konkurrenz produzierender Unternehmen aus dem Ausland unter Druck gesetzt und versucht mit Initiativen wie Industrie 4.0 (I4.0) durch die Vernetzung ihrer Betriebsinfrastruktur neue Geschäftsmodelle und flexiblere Prozesse zu schaffen, um variantenreiche Produkte schneller und zu geringeren Kosten zu produzieren. Einen Kernaspekt dazu stellt **Agilität** dar, mit der Unternehmen schnell auf Marktänderungen reagieren können. Schon zu Beginn des 21. Jahrhundert wurden hierzu Fertigungskonzepte entwickelt, mit denen **agile, rekonfigurierbare oder holonische Fertigungssysteme** entstehen sollten, die auf eine schnelle Wandlungsfähigkeit abzielen. Mit I4.0 sollen nun **Cyber-physische Produktionssysteme (CPPS)** entstehen, in denen die Agilität durch Internettechnologien zur Kommunikation und Interaktion zwischen den Maschinen und Komponenten eines Fertigungssystems unterstützt werden soll. Im Speziellen sollen mit dem Ziel der **agilen Automatisierung**, das mit dieser Arbeit verfolgt wird, Inbetriebnahme- und Rekonfigurationsaufwände von Maschinen in heterogenen, komplexen Fertigungssystemen reduziert werden. Dazu wird im Zusammenhang mit I4.0 häufig leichtsinnig von **semantischer Interoperabilität** gesprochen, wenn heterogene Maschinen miteinander kommunizieren und interagieren sollen. Dabei beruht der Grundgedanke hinter semantischer Interoperabilität grundsätzlich darauf, eine für eine Domäne einheitliche Sprache mit eindeutige Begriffen und Zusammenhängen zwischen diesen zu definieren, um damit letztendlich das altbekannte Standardisierungsproblem zu lösen.

Damit ist aber das grundlegende Interoperabilitätsproblem für Fertigungssysteme noch nicht gelöst. Denn Steuerungsansätze basierend auf semantischer Interoperabilität können selbst nur auf einer Menge vordefinierter statischer Regeln basieren, mit denen auf Grund der bekannten Maschinen und Komponenten und deren statischen Verhalten Entscheidungen getroffen werden können. Hierbei wird oft vernachlässigt, dass die Fähigkeiten eines gesamten Fertigungssystems in einer komplexen Abhängigkeit zueinander stehen, die durch diese vordefinierten Regeln abgebildet werden muss. Des Weiteren können in diesem Zusammenhang nur semantisch gleichartig beschriebene

Maschinen verwendet werden, jedoch sind bei technischen Systemen oft sehr spezifische Anwendungen notwendig, deren Verhalten nicht problemlos auf eine abstrakte Semantik reduziert werden kann.

Aus diesen Gründen wird in dieser Arbeit mit der Pragmatik auch die Beschreibung des maschinenspezifischen Verhaltens im Kontext der Anwendung berücksichtigt. Das hierin entwickelte Konzept beruht auf der Unterteilung der Semiotik (die in der Philosophie die Darstellung und Interpretation von Zeichen behandelt) in die Aspekte der Syntax, Semantik und Pragmatik, durch den Philosophen Charles W. Morris. Dabei wird das Wissen zum Verhalten der Maschinen, das bereits bei der Entwicklung und Programmierung in diese einfließt, in Form von orthogonalen Zustandsautomaten für die einzelnen Komponenten beschrieben und Abhängigkeiten zwischen diesen Komponenten in den Zustandsübergängen hinterlegt. Bei der Inbetriebnahme kann durch Auflösen der orthogonalen Zustandsautomaten ein einzelner Graph erzeugt werden, in dem Automatisierungsschritte zum Herstellen bestimmter Systemzustände dynamisch abgeleitet werden können. Dies erlaubt die Betrachtung einzelner Maschinen, wie auch von Fertigungszellen und -systeme, als Menge orthogonaler Zustandsautomaten. Auf dieser Ebene müssen jedoch auch die Abhängigkeiten zwischen den Maschinen und deren Komponenten berücksichtigt werden. Zu diesem Zweck werden Abhängigkeiten, sowie Schnittstellen zu anderen Maschinen, als externe Platzhalter ebenfalls in Form von Zustandsautomaten modelliert. Das vorgestellte Konzept wird anhand der Modellierung einer Werkzeugmaschine im Kontext einer Fertigungszelle überprüft und hinsichtlich der Ableitung der Automatisierungsschritte evaluiert.

Abstract

The European manufacturing industry is under pressure from increasing competition from manufacturing companies abroad and, through initiatives such as Industry 4.0 (I4.0), is trying to create new business models and more flexible processes through the interconenction of manufacturing equipment in order to produce variant-rich products faster and at lower cost. A core aspect of this is **agility**, which enables companies to respond quickly to market changes. Already at the beginning of the 21_{st} century, manufacturing concepts were developed for this purpose, with which **agile, reconfigurable or holonic manufacturing systems** were to be created, which aim at rapid changeability. With I4.0, **cyber-pysical production systems (CPPS)** are now to be created in which agility is to be supported by Internet technologies for communication and interaction between the machines and components of a manufacturing system. Specifically, the goal of **agile automation** pursued in this thesis is to reduce commissioning and reconfiguration efforts of machines in heterogeneous, complex manufacturing systems. To this end, **semantic interoperability** is often carelessly referred to in connection with I4.0, when heterogeneous machines are supposed to communicate and interact with each other. The basic idea behind semantic interoperability is to define a uniform language for a domain with unambiguous terms and relationships between them in order to ultimately solve the well-known standardization problem.

However, this does not solve the fundamental interoperability problem for manufacturing systems. This is because control approaches based on semantic interoperability can themselves only be based on a set of predefined static rules with which decisions can be made on the basis of the known machines and components and their static behavior. Here, it is often neglected that the capabilities of an entire manufacturing system posses complex dependencies among each other, which have to be represented within these predefined rules. Furthermore, only semantically identically described machines can be used in this context, but technical systems often require very specific applications whose behavior cannot be easily reduced to abstract semantics.

For these reasons, pragmatics is used in this thesis to include the description of

machine-specific behavior in the context of application. The concept developed herein is based on the division of semiotics (which in philosophy deals with the representation and interpretation of signs) into the aspects of syntax, semantics, and pragmatics, by the philosopher Charles W. Morris. Here, the knowledge about the behavior of the machines, which is already embedded into them during development and programming, is described in the form of orthogonal state machines for the individual components, while dependencies between these components are stored in the state transitions. During commissioning, a single graph can be generated by resolving the orthogonal state machines, in which automation steps for establishing specific system states can be derived dynamically. This allows for individual machines, as well as for manufacturing cells and systems, to be viewed as a set of orthogonal state machines. At this level, however, the dependencies between the machines and their components must also be considered. For this purpose, dependencies, as well as interfaces to other machines, are modeled as external placeholders also in the form of state machines. The presented concept is tested by modeling a machine tool in the context of a manufacturing cell and evaluated with respect to the derivation of the automation steps.

Inhaltsverzeichnis

Kurzfassung **ix**

Abstract **xi**

Inhaltsverzeichnis **xiii**

Abbildungsverzeichnis **xv**

Tabellenverzeichnis **xviii**

Codeabschnitte **xix**

Abkürzungsverzeichnis **xxi**

1 Europa und die Chancen der Digitalisierung für die Fertigung **1**
1.1 Entwicklung und Trends in der Produktion 1
1.2 Herausforderungen 4
1.3 Problemstellung 6
1.4 Struktur der Arbeit 7

2 Die Entwicklung zum agil-rekonfigurierbaren Fertigungssystem **9**
2.1 Konventionelle Fertigungssysteme 10
2.2 Progressive Fertigungskonzepte 12
2.2.1 Agile Fertigung 12
2.2.2 Rekonfigurierbare Fertigungssysteme 14
2.2.3 Holonische Fertigungssysteme 16
2.2.4 Plug & Produce (P&P) 17
2.2.5 Cyber-physische Produktionssysteme (CPPS) 19
2.3 Agil-rekonfigurierbare Fertigungssysteme 19

3 Die Grenzen der semantischen Interoperabilität **21**
3.1 Sichtweisen der Interoperabilität 21

3.1.1 Wissenspyramide . . . 22
3.1.2 Levels of Conceptual Interoperability Model (LCIM) . . . 24
3.1.3 Dimensionen der Interoperabilität nach ETSI und IERC . . . 25
3.1.4 Semiotic Interoperability Framework . . . 26
3.1.5 Interoperabilität nach ISO/IEC 21823-1 . . . 27
3.1.6 Interoperabilität im Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) . . . 28
3.1.7 Interoperabilität nach Industrial Internet Reference Architecture (IIRA) . . . 32
3.1.8 Zusammenfassung: Modelle zur Interoperabilität . . . 34
3.2 Semantische Interoperabilität . . . 35
3.2.1 Ontologien, Taxonomien & Vokabulare . . . 36
3.2.2 OPC Unified Architecture Companion Specifications . . . 37
3.3 Vom Semantic Web zum Pragmatic und Dynamic Web . . . 39
3.4 Pragmatische Interoperabilität . . . 41
3.4.1 Konzepte pragmatischer Modellierung . . . 42
3.4.2 Endliche Zustandsautomaten . . . 43
3.4.3 Skills . . . 48
3.4.4 Servicekomposition . . . 49
3.5 Die Entwicklung der Interoperabilität . . . 50

4 Implikationen im Rahmen der Arbeit 51
4.1 Forschungsfragen . . . 51
4.2 Hypothesen und Methoden . . . 52
4.3 Interoperabilität von technischen Systemen . . . 53
4.4 Semantik und Pragmatik als Grundlage der Interoperabilität . . . 55
4.4.1 Gedankenexperiment zur Semiotik . . . 56
4.4.2 Zusammenspiel von Syntax, Semantik und Pragmatik . . . 57
4.4.3 Semantische Interoperabilität komplexer Systeme . . . 57

5 Semiotische Interoperabilität agiler Fertigungszellen 61
5.1 Verteiltes Wissen und Wissensträger . . . 61
5.1.1 Wissenstransfer bei der klassischen Automatisierung von Fertigungszellen . . . 63
5.1.2 Wissenstransfer bei der agilen Automatisierung von Fertigungszellen . . . 64
5.2 Wissensaustausch durch Interoperabilität . . . 66
5.2.1 Technische und syntaktische Interoperabilität . . . 67
5.2.2 Semantische Interoperabilität . . . 67
5.2.3 Pragmatische Interoperabilität . . . 68
5.2.4 Dynamische Interoperabilität . . . 73

5.2.5 Konzeptuelle Interoperabilität 83
5.3 Der Weg zur agilen Automatisierung 85

6 Implementierung und Proof of Concept 87
6.1 Fertigungszelle 87
6.2 Werkzeugmaschine 89
6.2.1 ClampingSystem (CS) 90
6.2.2 SideDoor (SD) 91
6.2.3 FrontDoor (FD) 92
6.2.4 NumericalControl (NC) 93
6.2.5 AuxiliaryDrives (AD) 94
6.2.6 HandlingKeyswitch (HK) 95
6.2.7 EmergencyStop (ES) 96
6.2.8 OperationMode (OM) 96
6.2.9 TransferPosition (TP) 97
6.2.10 MachiningTable (MT) 98
6.3 Handhabungsroboter 99
6.3.1 RobotGripper (RG) 100
6.3.2 RobotLoad (RL) 101
6.3.3 WorkpieceHandling (WH) 102
6.4 Zusammenfassung 104

7 Evaluierung 105
7.1 Aufgaben, Ausgangszustände und Ziele 105
7.1.1 Beladen 106
7.1.2 Fertigen 107
7.1.3 Entladen 107
7.2 Auflösung und Validierung der Zustandsautomaten 107
7.3 Graphsuche und automatische Generierung von Steuerungsprozessen . . 109
7.3.1 Beladen 110
7.3.2 Fertigen 111
7.3.3 Entladen 112
7.4 Zusammenfassung 112

8 Semiotik als Lösung zur agilen Automatisierung? 115
8.1 Zusammenfassung 115
8.2 Ausblick 118

Literatur 121

Abbildungsverzeichnis

1.1 Produktvolumen vs. Produktvielfalt 2
1.2 Anteil der Fertigung am Bruttoinlandsprodukt in Europa 4
1.3 Technische Systeme und ihre Beziehungen 5

2.1 Konzepte der flexibel automatisierten Fertigung 10

3.1 Wissenspyramide 22
3.2 Sender-Empfänger-Modell 23
3.3 Schichten der Interoperabilität im Levels of Conceptual Interoperability Model (LCIM) 25
3.4 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) 29
3.5 Interaktionsmodell zwischen I4.0-Komponenten 31
3.6 Industrial Internet Reference Architecture (IIRA) vs. Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) 33
3.7 OPC UA Übersicht 38
3.8 Moore- vs. Mealy-Automat 43
3.9 Hierarchie in Zustandsautomaten 44
3.10 Hierarchie in Zustandsautomaten 45
3.11 Kommunikation zwischen orthogonalen Zuständen 46
3.12 Hierarchie und Orthogonalität in State Chart XML (SCXML) 46

4.1 Gründe für Inbetriebnahmeaufwände 54
4.2 Gedankenexperiment zur Interoperabilität durch Semiotik 56

5.1 Wissen, Wissensträger und -transfer bei der klassischen Automatisierung einer Fertigungszelle 62
5.2 Wissenstransfer zur Zellensteuerung bei der klassischen Automatisierung 63
5.3 Wissen, Wissensträger und -transfer bei der agilen Automatisierung einer Fertigungszelle 65
5.4 Bausteine der semiotischen Interoperabilität agiler Fertigungszellen . . 66
5.5 Beispiel für die pragmatische Beschreibung der Zustände, Zustandsübergänge und deren Abhängigkeiten untereinander 69

5.6 Zustandsautomat aus einem Programmabschnitt einer Speicherprogrammierbaren Steuerung (Speicherprogrammierbare Steuerung (SPS)) . . . 72
5.7 Formale Beschreibung von hierarchischen Zustandsautomaten 76
5.8 Formale Beschreibung von orthogonalen Zustandsautomaten 77
5.9 Formale Beschreibung von transformierten orthogonalen Zustandsautomaten 78
5.10 Vereinfachte Darstellung von transformierten orthogonalen Zustandsautomaten 78
5.11 Umwandlung in einen flachen Zustandsautomaten 80
5.12 Validierung der Zustandsübergänge 81

6.1 Fertigungszelle 88
6.2 Maschinen und Subkomponenten 88
6.3 Abhängigkeiten zwischen den Komponenten der Werkzeugmaschine . . 89
6.4 Zustandsautomat ClampingSystem (CS) 90
6.5 Zustandsautomat SideDoor (SD) 91
6.6 Zustandsautomat FrontDoor (FD) 92
6.7 Zustandsautomat NumericalControl (NC) 93
6.8 Zustandsautomat AuxiliaryDrives (AD) 94
6.9 Zustandsautomat HandlingKeyswitch (HK) 95
6.10 Zustandsautomat EmergencyStop (ES) 96
6.11 Zustandsautomat OperationMode (OM) 96
6.12 Zustandsautomat TransferPosition (TP) 97
6.13 Zustandsautomat MachiningTable (MT) 98
6.14 Abhängigkeiten zwischen den Komponenten des Handhabungsroboters . 100
6.15 Zustandsautomat RobotGripper (RG) 100
6.16 Zustandsautomat RobotLoad (RL) 101
6.17 Zustandsautomat WorkpieceHandling (WH) 102

7.1 GRAFCET Steuerungsprozess zur Beladung 110
7.2 GRAFCET Steuerungsprozess zur Bearbeitung 112
7.3 GRAFCET Steuerungsprozess zur Entladung 113

Tabellenverzeichnis

5.1 Subzustände in orthogonalen Zuständen 69
5.2 Zustandsübergänge in orthogonalen Zuständen 70
5.4 Interne Signale und aggregierte Zustände 71

6.1 Abhängigkeiten von ClampingSystem (CS) 90
6.2 Abhängigkeiten von SideDoor (SD) 92
6.3 Abhängigkeiten von FrontDoor (FD) 93
6.4 Abhängigkeiten von NumericalControl (NC) 94
6.5 Abhängigkeiten von AuxiliaryDrives (AD) 95
6.6 Abhängigkeiten von OperationMode (OM) 97
6.7 Abhängigkeiten von TransferPosition (TP) 98
6.8 Abhängigkeiten von MachiningTable (MT) 99
6.9 Abhängigkeiten von WorkpieceHandling (WH) 103

7.1 Ausgangszustände der Einzelkomponenten beim Beladen, Fertigen und Entladen 106
7.2 Ausführung von graken 108

Codeabschnitte

3.1 Hierarchie von Zuständen in State Chart XML (SCXML). 47
3.2 Parallele Zustände in State Chart XML (SCXML). 48
5.1 Interne Signale und Bedingungen im Code der Speicherprogrammierbaren Steuerung (SPS) 72

Abkürzungsverzeichnis

AD AuxiliaryDrives
AUML Agent Unified Modeling Language
BFS Breadth First Search
BIP Bruttoinlandsprodukt
BPEL Business Process Execution Language
CCXML Call Control XML
CNC Computerized Numerical Control
CPPS Cyber-physische Produktionssysteme
CPS Cyber-physische Systeme
CS ClampingSystem
DFS Depth First Search
DIN Deutsches Institut für Normung
DKE Deutsche Kommission Elektrotechnik
ES EmergencyStop
ETSI European Telecommunications Standards Institute
FD FrontDoor
FSM Finite State Machine
HK HandlingKeyswitch
I4.0 Industrie 4.0
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronics Engineers
IERC European Research Cluster on the Internet of Things
IIC Industrial Internet Consortium
IIoT Industrial Internet of Things
IIRA Industrial Internet Reference Architecture
IKT Informations- und Kommunikationstechnologien
IoT Internet of Things
IoTS Internet of Things and Services
ISO International Organization for Standardization
KMU Kleine und mittlere Unternehmen
LCIM Levels of Conceptual Interoperability Model
MAS Multiagentensystem

MES Manufacturing Execution System
MS Modellierung und Simulation
MT MachiningTable
NC NumericalControl
OPC OLE for Process Control
OM OperationMode
OMG Object Management Group
OPC Open Platform Communications
OPC DA OPC Data Access
OPC UA OPC Unified Architecture
OWL OWL Web Ontology Language
OWL-S OWL Web Ontology Language for Services
PLM Product Lifecycle Management
PnP Plug & Play
P&P Plug & Produce
PPR Produkt-Prozess-Ressource
RAMI4.0 Referenzarchitekturmodell Industrie 4.0
RDF Resource Description Framework
RDFS RDF Schema
RG RobotGripper
RL RobotLoad
SCXML State Chart XML
SD SideDoor
SPS Speicherprogrammierbare Steuerung
SysML Systems Modeling Language
TP TransferPosition
umati universal machine technology interface
UML Unified Modelling Language
URI Unique Resource Identifier
URL Unique Resource Locator
VDE Verband der Elektrotechnik, Elektronik und Informationstechnik
VDI Verein Deutscher Ingenieure
VDMA Verband Deutscher Maschinen- und Anlagenbau
WH WorkpieceHandling
XML eXtensible Markup Language
ZVEI Zentralverband Elektrotechnik- und Elektronikindustrie

Kapitel 1

Europa und die Chancen der Digitalisierung für die Fertigung

Die europäische Fertigungsindustrie ist von der zunehmenden Konkurrenz produzierender Unternehmen aus dem Ausland unter Druck gesetzt und versucht mit Initiativen wie Industrie 4.0 (I4.0) durch die Vernetzung ihrer Betriebsinfrastruktur neue Geschäftsmodelle und flexiblere Prozesse zu schaffen, um variantenreiche Produkte schneller und zu geringeren Kosten zu produzieren.

1.1 Entwicklung und Trends in der Produktion

Schon gegen Ende des 20. Jahrhunderts hat sich der rückläufige Trend der Massenfertigung von hohen Produktvolumina hin zu variantenreichen und individuellen Einzelprodukten und einer damit höheren Produktvielfalt bemerkbar gemacht (Abbildung 1.1). Um mit diesen Anforderungen mithalten zu können, wurden neue Fertigungskonzepte entwickelt, die, ausgehend von flexiblen Transferstraßen, über flexible Fertigungssysteme, hin zu schnell anpassungsfähigen und rekonfigurierbaren Fertigungssystemen gehen. Dies hat sich zuerst mit teils abstrakten Konzepten, wie agilen Fertigungssystemen [4], über praktischere Konzepte, wie rekonfigurierbare Fertigungssysteme [5], holonische Fertigungssysteme und Plug & Produce (P&P) [6], hin zum heutigen Trend der Cyber-physischen Systeme (CPS) in der Produktion, den Cyber-physischen Produktionssystemen (CPPS) [7], entwickelt. Alle Konzepte haben gemein, dass mit ihnen nicht mehr nur die Flexibilität, sondern auch die Wandelbarkeit [8] des Fertigungssystems

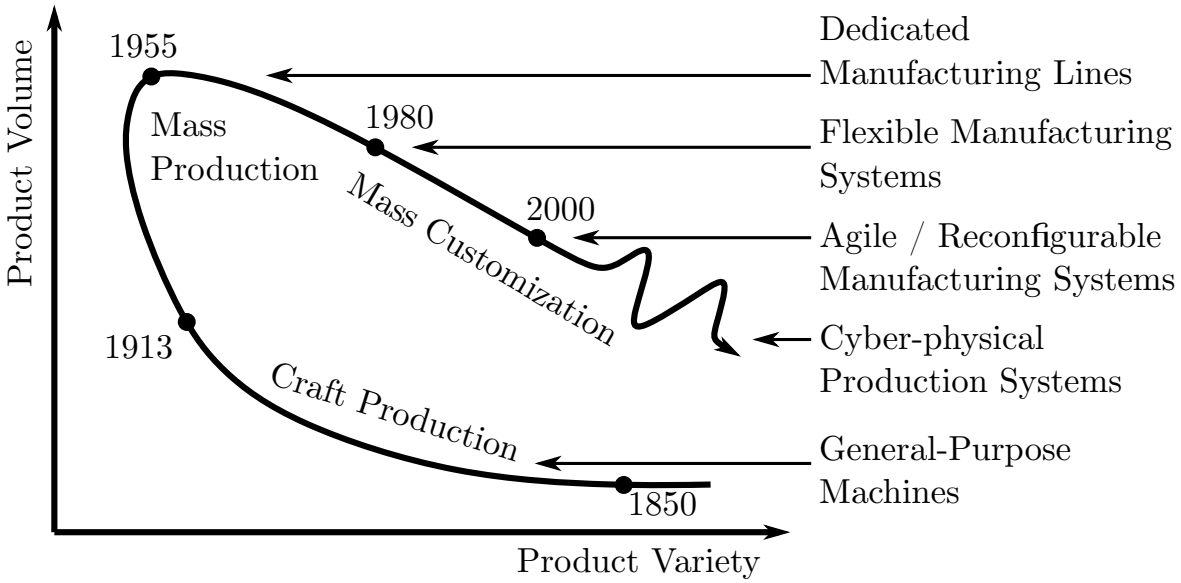


Abbildung 1.1: Produktvolumen vs. Produktvielfalt nach Jovane et al. [9] und Koren [10] und Erweiterung um agile Fertigung und Cyber-physische Produktionssysteme (CPPS).

und damit auch die Agilität im Vordergrund steht.

Als Grundlage für die Entwicklung zu CPPS wird die zunehmende Verschmelzung von physischer und digitaler Welt gesehen, die auch im Bereich der Produktion Einzug gehalten hat. Im Jahr 2011 wurde die Initiative I4.0 durch die Deutsche Bundesregierung, dem Bundesministerium für Bildung und Forschung und dem Bundesministerium für Wirtschaft und Energie, als Teil der Hightech-Strategie 2020[1] für die nachfolgenden 10 bis 15 Jahre ins Leben gerufen [11]. Mit I4.0 sollen die Forschung und Entwicklung in Richtung der Digitalisierung der Industrie schneller vorangetrieben und dadurch die Wettbewerbsfähigkeit durch technologische Innovation langfristig gesichert werden. Auf europäischer Ebene hat sich die Wirtschaft nach dem Einbruch des Bruttoinlandsprodukts (BIP) mit der Wirtschaftskrise 2009 nur langsam erholt. In einer Aussendung der Europäischen Kommission [12] setzte sich diese 2012 das Ziel, die sinkende Rolle der Industrie in Europa wieder zu steigern und den Anteil der Produktion am BIP Europas bis zum Jahr 2020 auf 20 % zu steigern, da die Industrie die Basis für den Reichtum und Erfolg der europäischen Ökonomie bildet. Dabei haben es die deutsche und osteuropäische Industrie geschafft, lange Zeit ihre Marktanteile zu erhöhen, während Länder wie Großbritannien und Frankreich einen starken Rückgang erlebt haben [13]. Laut Blanchet [13] kann sich die europäische Wirtschaft durch

[1] https://www.hightech-strategie.de/

eine starke Position in I4.0, trotz der dafür notwendigen Kapitalinvestitionen, wieder weiter nach vorne arbeiten. Kleine und mittlere Unternehmen (KMU) haben laut Roth [14] jedoch häufig Vorbehalte gegen frühe Investitionen in neue Technologien, da sich diese häufig erst sehr spät auszahlen würden. Nach Hirsch-Kreinsen und ten Hompel [15] werden sich diese Technologien vorerst kaum unter den weniger technologieintensiven KMU durchsetzen und auch in Großserien produzierende Unternehmen werden demgegenüber zurückhaltend sein. Die Autoren erwarten, dass sich vorerst nur technologieintensive mittelständische Firmen dahingehend mit der Umsetzung von I4.0 Technologien beschäftigen werden, da sie wegen ihren hohen Flexiblitätsanforderungen dazu gezwungen sind sich schnell anzupassen. Besonders Zulieferunternehmen sollten sich dabei mit dem Thema I4.0 auseinandersetzen, da Unternehmen, die zusammen in einer Wertschöpfungskette stehen, durch die zunehmende Vernetzung unter Druck geraten [14]. Sich als Unternehmen früh mit dem Thema auseinanderzusetzen, steigert langfristig die Wahrscheinlichkeit in I4.0 erfolgreich zu sein [16]. Dafür bieten sich mit der Digitalisierung in der Industrie einige Chancen für Unternehmen, die bereit sind, technologische Investitionen zu tätigen:

1. **Individualisierung** von Produkten durch die Realisierung individueller und kurzfristiger Anforderungen hinsichtlich Design, Planung und Produktion [14] wird besonders durch die Senkung der Komplexitätskosten [17] ermöglicht und erlaubt es stark individualisierte Produkte bis zur Stückzahl eins [18] wirtschaftlich rentabel zu erzeugen.

2. **Flexibilisierung und Agilität** erlauben eine Verkürzung der Lead-Time und Time-to-Market durch die durchgängige Integration von Product Lifecycle Management (PLM) Software auf horizontaler Ebene und eine schnelle Reaktionsfähigkeit auf Marktänderungen durch die dynamische Gestaltung von Produktionsprozessen in CPPS [14]. Die wachsende Leistung, Vernetzung, Dezentralisierung und Autonomie der Produktion erlaubt es mit der steigenden Komplexität zurechtzukommen, so dass Veränderungen des Systems teilweise zur Laufzeit erfolgen können [18].

3. Eine **Produktivitätssteigerung** durch dezentrale und autonome CPS zur Produktion [18] mit optimiertem Ressourcenverbrauch, prädiktiver Instandhaltung, und durchgängiger Simulation der Geschäftsprozesse [14].

4. Wirtschaftlichkeitspotenziale durch **Wertschöpfungsnetzwerke**, in denen Unternehmen mit Zulieferfirmen und Kundschaft hierarchiearm vernetzt sind [17] und eine verteilte Fertigung über mehrere Unternehmensstandorte oder unterschiedliche Unternehmen hinweg durch web-basierte Plattformen und die Verteilung von Arbeitsgängen ermöglicht wird [19]. Dadurch wird besonders die Eintrittsbarriere für KMU in den Markt gesenkt [13].

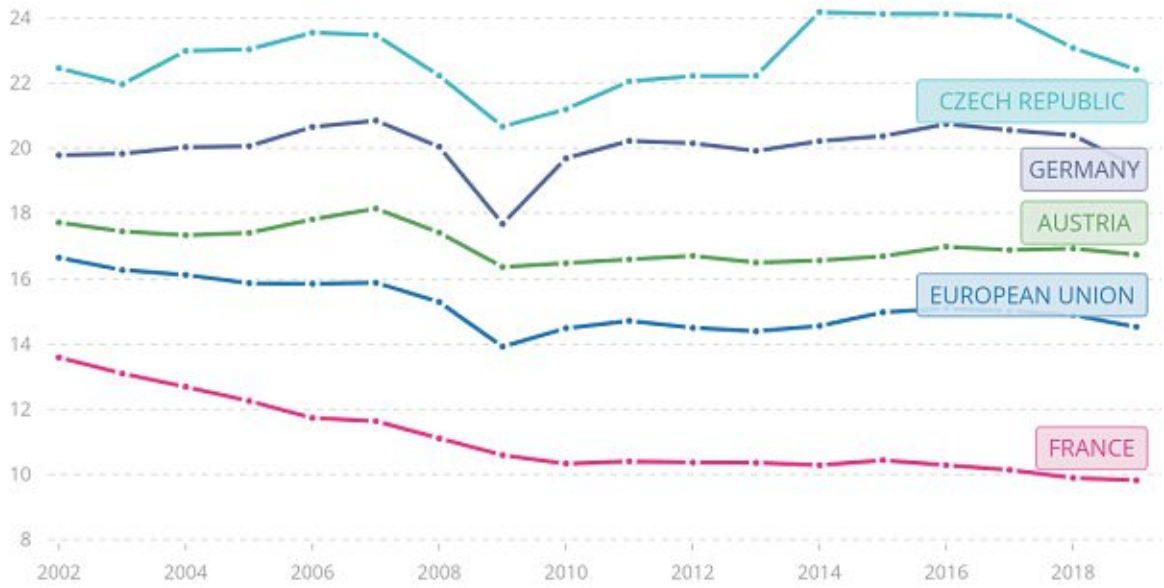


Abbildung 1.2: Anteil der Fertigung am BIP in Europa (2002 - 2019 in %) [20], lizensiert unter CC-4.0 BY.

1.2 Herausforderungen

Die Strategie der Europäischen Kommission, das BIP bis ins Jahr 2020 auf 20 % anzuheben [12], wurde bereits früh kritisiert, denn dazu müssten sich besonders Länder mit einem geringen Anteil, wie Frankreich, in denen die Industrie über viele Jahre hinweg stark zurückgegangen ist, in einem kurzen Zeitraum erholen [13]. Der Verlauf des BIP in Europa (Abbildung 1.2) hat sich unterdessen bis 2019 nahezu konstant verhalten, mit einem leichten Einbruch in 2019, der besonders die Automobilindustrie in Deutschland und deren Zulieferfirmen in Osteuropa getroffen hat. Es ist wohl kaum zu erwarten, dass sich der Anteil der Fertigung am BIP in Europa in den nächsten Jahren überdurchschnittlich erhöhen wird, um die 20 % zu erreichen.

Die Chancen der Digitalisierung haben sich bis jetzt noch nicht auf das BIP ausgewirkt. Hirsch-Kreinsen und ten Hompel [15] beschreiben, dass es sich bei I4.0 bisher noch um eine technische Vision handelt, die schwere ökonomische, soziale und komplexe technische Barrieren überwinden muss. Dastbaz [21] kritisiert, dass das Thema I4.0 zwar in den letzten Jahren viel Interesse in Industrie und Forschung geweckt hat, aber das Konzept und Potential von I4.0 oft nicht richtig verstanden wird, schlecht umgesetzt wird und es früher oft als Vorwand genutzt wurde, um alte Produkte un-

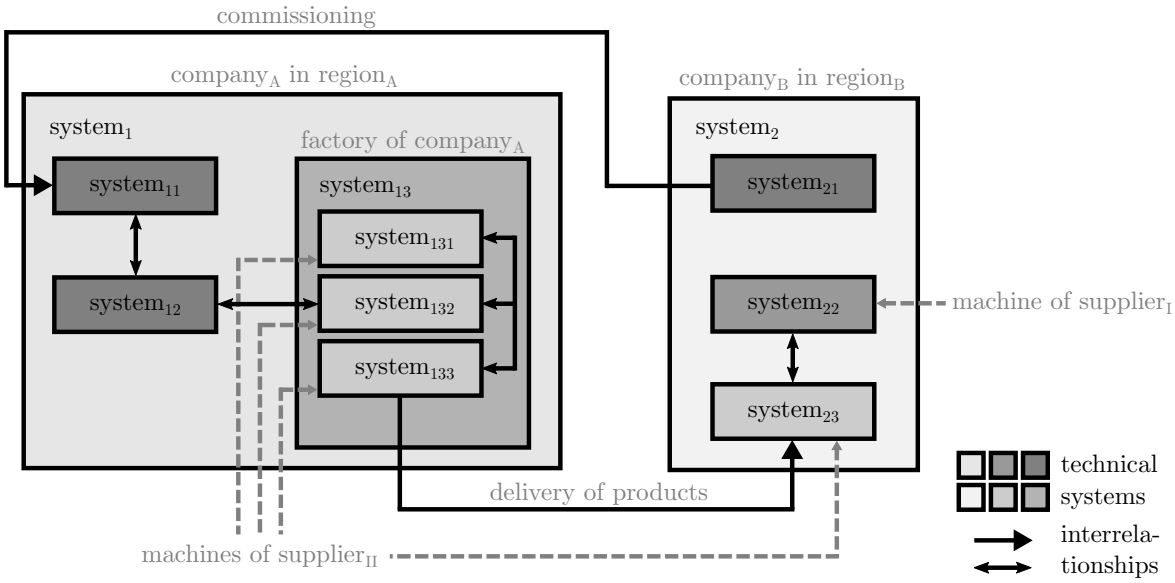


Abbildung 1.3: Technische Systeme und ihre Beziehungen untereinander [23].

ter einem neuen Deckmantel zu vertreiben. Bis zur selbst-organisierenden Produktion sind noch viele Entwicklungen in Forschung und Standardisierung notwendig [22]. Eine Umsetzung von I4.0 in einem volkswirtschaftlich bedeutenden Umfang ist daher nicht früher als im Jahr 2025 zu erwarten [16].

Eine der Kernherausforderungen mit denen sich Industrie und Forschung derzeit konfrontiert sehen, ist die Interoperabilität, also den Informationsaustausch und das Nutzen dieser Informationen [24], zwischen heterogenen und komplexen technischen Systemen (Abbildung 1.3) herzustellen, damit aus der durchgängigen Vernetzung überhaupt Vorteile gewonnen werden können. Die Komplexität technischer Systeme kommt daher, dass diese wiederum aus weiteren komplexen technischen Subsystemen bestehen, die auf unterschiedliche Weisen miteinander in Beziehung stehen. Nach Zeid et al. [25] muss, durch die Notwendigkeit der vertikalen und horizontalen Kommunikation die traditionell hierarchische Automatisierungspyramide in ein heterarchisches Netzwerk umgewandelt werden, und dazu besteht die Notwendigkeit der Standardisierung der syntaktischen und semantischen Interoperabilität. Auf Anlagenebene versucht man dies durch die Schaffung einheitlicher Schnittstellen, wie OPC Unified Architecture (OPC UA) [26], und durch die Interoperabilität semantischer Datenmodelle zu erreichen. In OPC UA Arbeitsgruppen [27] werden von Domänenexperten und Industrievertretern solche gemeinsamen Standards in Form von Begleitspezifikationen erarbeitet, um zu einer semantischen Interoperabilität zwischen Geräten unterschiedlicher Herstellerfirmen kommen.

1.3 Problemstellung

Semantische Interoperabilität benötigt einheitliche Modelle und eine einheitliche Bedeutung für die darin definierten Konzepte. Werden diese semantischen Beschreibungen von mehreren Parteien gemeinsam in Ontologien definiert, verwendet und weiterentwickelt, dann sind diese untereinander interoperabel [28]. Semantische Interoperabilität benötigt damit Standardisierung im industriellen Umfeld. Als Folge müssen sich Unternehmen herstellungs- und anwendungsseitig in einer Domäne absprechen und Datenmodelle mit einheitlicher Struktur und Semantik liefern.

Zur reinen Datenerfassung von Anlagen ist es verhältnismäßig einfach einzelne Datenpunkte semantisch zu beschreiben, so lange sich mehrere Parteien auf ein einheitliches Modell einigen können. Zur Steuerung von Anlagen im Sinne von P&P ist das wesentlich aufwändiger, da die Auswirkungen von einzelnen Funktionen der Anlage auch herstellerübergreifend einheitlich sein müssten, damit diese Funktionen mit einer standardisierbaren Steuerungslogik funktionieren. Da die angebotenen Funktionen und deren Auswirkungen stark von den Beziehungen zwischen den technischen Systemen abhängen, entspricht die Komplexität zur Beschreibung solcher Systeme auch der Komplexität der Systeme selbst. Die Entwicklung standardisierter Semantik ist folglich dort einfach, wo Maschinen eine geringe Komplexität und hohe Ähnlichkeit besitzen (zum Beispiel zur Standardisierung einfacher Sensoren, wie Temperatursensoren), ist aber schwierig, wo Maschinen eine höhere Komplexität besitzen (zum Beispiel technische Systeme wie Fertigungszellen). Bei der Steuerung von Systemen ergeben sich daher trotz semantischer Interoperabilität die folgenden Probleme:

1. Semantische Interoperabilität geht nur von allgemein in der Ontologie bekannten Konzepten aus. Auf neue Konzepte muss man sich erst in der Gemeinschaft einigen, was die Verwertung von proprietären Eigenschaften und Fähigkeiten bei der Beschreibung von Geräten behindert, da der Fokus auf Standardisierung liegt.

2. Semantische Interoperabilität erlaubt zwar eine standardisierte Steuerung, dabei muss aber die pragmatische Komponente, also die Funktion und deren Auswirkung auf das System, auch standardisiert und der Steuerungslogik im Vorhinein bekannt sein.

3. Selbst wenn die Einzelkomponenten eines komplexen Systems semantisch beschrieben sind und deren Funktionen standardisiert sind, können daraus nicht automatisch die Auswirkungen dieser Funktionen auf ein komplexes System inferiert werden.

Es müssen folglich Lösungen gefunden werden, um einen Nutzen aus semantischer Interoperabilität zur automatisierten Konfiguration und Steuerung komplexer Systeme ziehen zu können. In dieser Arbeit werden dazu vorhandene Ansätze untersucht, ein Lösungsansatz entwickelt und mit der Umsetzung in einer Fertigungszelle evaluiert.

1.4 Struktur der Arbeit

Im folgenden Kapitel werden die aktuellen Forschungsansätze im Bereich von Fertigungssystemen den konventionellen Ansätzen gegenübergestellt und der Trend in Richtung von agilen und anpassungsfähigen Fertigungssystemen hervorgehoben. In Kapitel 3 werden die unterschiedlichen, teils technischen, teils philosophischen, Ansätze zur Interoperabilität betrachtet, und dabei der Zusammenhang zwischen der Semantik und der Pragmatik aufgezeigt. Aus den erarbeiteten Inhalten werden in Kapitel 4 die Grundannahmen und Hypothesen abgeleitet. Auf dieser Basis wird im darauf folgenden Kapitel 5 das Konzept für die Semiotische Interoperabilität agiler Fertigungszellen entwickelt. Mit dem Proof of Concept in Kapitel 6 folgt die beispielhafte Umsetzung zweier Teilaspekte des erarbeiteten Konzepts durch die Modellierung der Pragmatik einer Werkzeugmaschine innerhalb einer Fertigungszelle, für die anschließende Evaluierung hinsichtlich einer agilen Automatisierung in Kapitel 7.

Kapitel 2

Die Entwicklung zum agil-rekonfigurierbaren Fertigungssystem

Steigende Nachfrage nach individuellen Produkten mit kleineren Losgrößen und kurzfristig schwankende Märkte machen produzierenden Unternehmen zu schaffen. Schurig et al. [29] nennen verschiedene Konzepte aus der Literatur, mit denen sich Produktionsunternehmen auf die zunehmende Volatilität des Marktes und Unsicherheiten vorbereiten können: Flexibilität, Wandlungsfähigkeit, Robustheit, Resilienz oder Agilität. Diese Konzepte lassen insgesamt einen Trend erkennen, in dem Schwierigkeiten durch schwankende Nachfrage und individuelle Produkte durch Konzepte für eine agile und anpassungsfähige Produktion und Organisation gelöst werden sollen. Laut Duden bedeutet das Adjektiv agil "von großer Beweglichkeit zeugend; regsam und wendig" [30]. Im Kontext einer agilen Fertigung handelt es sich folglich um eine Fertigung, die sehr anpassungsfähig und schnell implementiert und konfiguriert werden kann. Dabei spielen aktuelle Entwicklungen und Technologien im Bereich der Digitalisierung der Produktion wie Internet of Things (IoT), Industrial Internet of Things (IIoT) und I4.0 als *Enabler* dieser Vision eine große Rolle.

2.1 Konventionelle Fertigungssysteme

Konventionelle Fertigungssysteme lassen sich über die beiden Größen der Produktivität und der Flexibilität einteilen und haben das Problem, dass Flexibilität im notwendigen Ausmaß oft schwierig mit einem hohen Automatisierungsgrad und hoher Produktivität vereinbar ist. Michel und Kraushaar [31] unterscheiden sechs Arten von Konzepten der flexibel automatisierten Fertigung (Abbildung 2.1):

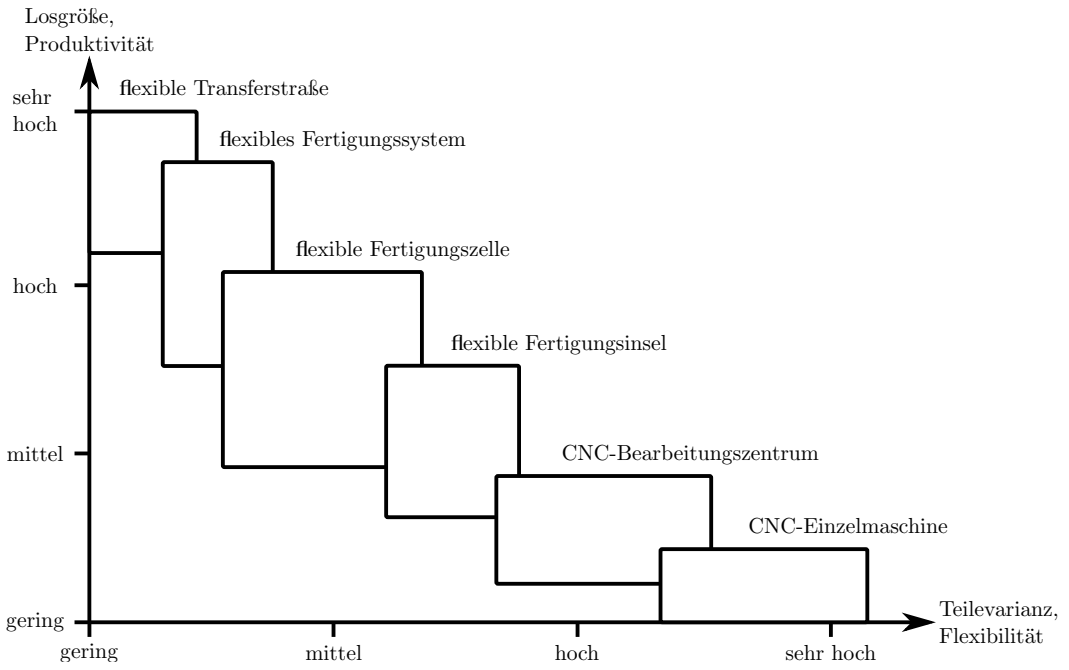


Abbildung 2.1: Konzepte der flexibel automatisierten Fertigung von der Einzelmaschine mit numerischer Steuerung (CNC) bis zur flexiblen Transferstraße [31].

- Die **CNC-Einzelmaschine** ist laut den Autoren [31] der Ausgangspunkt einer flexiblen Fertigung und hat in der Regel noch keine automatischen Werkstückwechseleinrichtungen. Die einfachste Form einer Fräsmaschine besteht laut Glockner [32] aus einem festen Maschinentisch mit drei Verfahrachsen, die von einer **numerischen Steuerung** (engl. Computerized Numerical Control (CNC)) kontrolliert werden. Eine CNC-Einzelmaschine besitzt damit eine höhere Wiederholgenauigkeit, hohe Verfahrgeschwindigkeiten und damit einhergehend auch eine höhere Produktivität als manuell gesteuerte Maschinen.
- Im Gegensatz zur CNC-Einzelmaschine, ist ein **CNC-Bearbeitungszentrum** dadurch charakterisiert, dass **mehrere Bearbeitungsverfahren** in eine Maschine integriert sind [31]. Laut Fechter et al. [22] spricht man bereits von einem

Bearbeitungszentrum, wenn zu einer CNC-Maschine ein **Werkzeugwechsler** mit **Werkzeugmagazin** ergänzt wird, da die Produktivität durch die Parallelisierung von Haupt- und Nebenzeiten erheblich gesteigert wird und die Nebenzeiten gesenkt werden. Zusätzlich kommen heutzutage oft auch schon **Werkstück-** bzw. **Palettenwechsler** zum Einsatz, die eine hauptzeitparallele Werkstückspannung ermöglichen.

- Die **flexible Fertigungsinsel** ist keine Weiterentwicklung im Sinne der Automatisierung, sondern ein arbeitstechnisches Konzept, in dem Arbeitsplätze zur Fertigbearbeitung einer Werkstückfamilie räumlich und organisatorisch zusammengefasst werden [31].

- Eine **flexible Fertigungszelle** ist nach Michel und Kraushaar [31] und Fechter et al. [22] im Gegensatz zur CNC-Einzelmaschine oder zum CNC-Bearbeitungszentrum durch zusätzliche Einrichtungen wie **Werkstückspeicher, Werkstückwechseleinrichtung, Werkzeugüberwachung, Reinigungsfunktion und Bearbeitungs- und Qualitätskontrolle** in der Lage, eine begrenzte Zeit bedienerlos zu arbeiten und dabei unterschiedliche Werkstücke automatisch zu bearbeiten. Im Werkstückspeicher werden die Werkstücke entweder vereinzelt oder auf einzelnen Paletten mit Spannvorrichtungen mit entweder manueller oder automatischer Bedienung gespannt und für die Bearbeitung bereitgehalten. Fertigbearbeitete Teile werden entweder in einem weiteren oder im selben Speicher zwischengelagert.

- In einem **flexiblen Fertigungssystem** sind mehrere CNC-gesteuerte Bearbeitungsmaschinen zur Bearbeitung von Werkstück- und Teilefamilien zusammengefasst, die unabhängig voneinander arbeiten und die Komplettbearbeitung von Rohteilen und Halbzeugen erlauben [31]. Nach Fechter et al. [22] wird der **Werkzeug- und Werkstückaustausch** über einen **Fertigungsleitrechner** gesteuert und erfolgt automatisch. Mit einem flexiblen Materialfluss zwischen den Einzelmaschinen verkürzen sich außerdem Durchlaufzeiten für die Komplettbearbeitung unterschiedlicher Werkstücke und eine hohe Produktivität mit gleichzeitig hoher Flexibilität wird erreicht. Der nicht richtungsgebundene Materialfluss zwischen den Bearbeitungsmaschinen wird über **Fördersysteme, Werkstückspeicher oder Handhabungsgeräte** durchgeführt [31] und erfordert eine komplexe Steuerung, die rechtzeitig Rohteile, Werkzeuge und Vorrichtungen bereitstellt [22].

- Eine **flexible Transferstraße** ist ein Fertigungssystem, das aus linear mit einen Werkstücktransportsystem verketteten Fertigungsmaschinen besteht und gleichzeitig oder sequentiell verschiedene Werkstücke bearbeiten kann [31]. Flexible Transferstraßen haben kurze Durchlaufzeiten und sind besonders für geometrisch

ähnliche Teilefamilien geeignet, die längerfristig konstante Produktionszahlen haben [22]. Die Voraussetzung hierfür ist eine Typenreihe mit ähnlichen Arbeitsvorgangsfolgen [31].

Von den Konzepten der flexibel automatisierten Fertigung (Abbildung 2.1) sind die flexiblen Systeme von der CNC-Einzelmaschine bis zum flexiblen Fertigungssystem vorwiegend bei kleineren bis mittleren Losgrößen anzutreffen und in Bereichen, die stark durch die Anforderung an Produktvielfalt durch Auftragsfertigung geprägt sind. Die flexible Transferstraße ist aufgrund der eingeschränkten Flexibilität in der Produktvielfalt eher in der Großserienfertigung anzutreffen, wo eine sehr hohe Produktivität gefordert ist [33].

2.2 Progressive Fertigungskonzepte

Nach Bauer [33] sind in der Fertigung Automatisierung und Flexibilisierung konkurrierende Zielgrößen. Ein höherer Automatisierungsgrad schränkt folglich die Flexibilität durch fixierte Prozesse und Produktfamilien ein. Ein hoher Grad an Flexibilität lässt sich wiederum schwierig automatisieren. Mit steigender Automatisierung steigt aber häufig auch die Produktivität von Produktionsanlagen. Mit der Zunahme der Nachfrage nach individuellen Produkten und kleinen Losgrößen wird aber die Notwendigkeit von Flexibilität im Sinne der individuellen Produktion gesteigert, weshalb Flexibilisierung ebenso erstrebenswert wie Automatisierung ist. Flexible Fertigungssysteme sind zwar bereits darauf optimiert, ein möglichst breites Spektrum an Werkstücken in unterschiedlichster Reihenfolge zu bearbeiten, haben aber hohe Investitionskosten und benötigen einen besonders hohen Detaillierungsgrad und Erfahrung in der Arbeitsvorbereitung. KMU haben es hiermit besonders schwer, aber die Entwicklungen aus IoT und I4.0 sollen besonders auch KMU helfen die Flexibilität und Produktivität zu steigern. Im Laufe der Zeit haben sich daher unterschiedlichste Konzepte für Fertigung entwickelt.

2.2.1 Agile Fertigung

Der Begriff agile Fertigung wurde im Jahr 1991 von Nagel und Dove [4] im Zusammenhang mit agilen Unternehmen geprägt, um mit der zunehmenden Abnahme der Massenproduktion die Vision für das kommende Jahrhundert zu schaffen. Laut den Autoren sind agile Unternehmen dabei komplett integrierte Organisationen, in denen

Informationen durchgängig zwischen Organisationseinheiten ausgetauscht werden und Arbeit eher simultan, als sequentiell vorangeht. Diese zeichnen sich aus durch:

- die Möglichkeit zum Austausch von Informationen fördert die Entstehung von "virtuellen" Unternehmen
- Produktionsnetzwerke und Kooperation zwischen Zulieferfirmen, Herstellerfirmen und Kundschaft
- modulare und kompatibel miteinander austauschbare Produktions- und Organisationseinheiten
- gemeinsam genutzte Produktionsressourcen zwischen Unternehmen
- eine durchgängige Prozesskontrolle durch Überwachung, Analyse und Regelkreise
- Integration von Feedback der anwendenden Personen, das es erlaubt schnell zu reagieren und diese Informationen schnell umzusetzen
- intensive Prozessmodellierung und Simulation, die es erlauben Vorversuche zu eliminieren und dadurch Zeit und Kosten zu sparen

Nach Yusuf et al. [34, S. 37] ist Agilität "die erfolgreiche Erkundung wettbewerbsfähiger Grundlagen (Geschwindigkeit, Flexibilität, Innovationsaktivität, Qualität und Rentabilität) durch die Integration rekonfigurierbarer Ressourcen und Best Practices in eine wissensreiche Umgebung, um publikumsgetriebene Produkte und Dienstleistungen in einem sich schnell ändernden Markt bereitzustellen."[2] Laut den Autoren müssen Fertigungsunternehmen Produkte kostengünstig, qualitativ hochwertig und schnell produzieren und dabei proaktiv und innovativ sein, um wettbewerbsfähig zu bleiben. Agile Unternehmen müssen mit Maßnahmen zur Reaktion proaktiv auf Veränderungen vorbereitet sein, um schnell reagieren zu können. Für diese Maßnahmen sind weitreichende Änderungen über die gesamte Wertschöpfungskette möglich, weshalb Flexibilität und Wandlungsfähigkeit integrale Bestandteile sind [29]. Laut Nagel und Dove [4] sind hochflexible Fertigungsanlagen zwar notwendig für die agile Fertigung, aber nicht ausreichend, da auch Management und Mitarbeiter eine Rolle spielen. Laut Berger [35] ist es für die Anlagenproduktivität in einem agilen Fertigungssystem wichtig, dass die Maschinenbetreiber ein hohes Maß an Qualifikation mitbringen, weshalb auch Schulungskapazitäten notwendig sind.

[2] "Agility is the successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environments" [34, S. 37].

Im agilen Fertigungssystem muss im Gegensatz zum sequentiellen Ablauf von Arbeitsschritten, wie bei der flexiblen Transferstraße, ein paralleles Abarbeiten von unterschiedlichen Operationen möglich sein, das dem Ansatz eines flexiblen Fertigungssystems ähnelt. Zusätzlich muss es aber im Sinne der Agilität möglich sein kurzfristig auf neue Marktanforderungen, Produkte und Fertigungstechnologien, sowie Absatzschwankungen, reagieren zu können. Eine Rekonfiguration des Fertigungssystems, ein Austausch von Maschinen oder eine Erweiterung um einzelne Maschinen muss kurzfristig möglich sein. Plug & Produce [6] oder sogar Rent & Produce [36] Ansätze sind denkbar, um kurzfristig derartige schnelle Änderungen zu bewirken. Auch Ansätze von agilen Fertigungssystemen durch eine flexible Transportlösung mit fahrerlosen Transportfahrzeugen sind denkbar, wenn dadurch Fertigungskapazitäten aufgrund der losen Kopplung der Maschinen schnell durch den Einsatz weiterer Maschinen angepasst werden können. Laut Nagel und Dove [4] benötigt die agile Fertigung dazu eine Reihe von Technologien:

- flexible, programmierbare Werkzeugmaschinen, die in rekonfigurierbaren und modularen Fertigungszellen organisiert sind,
- "intelligente" Steuerungen für Fertigungsprozesse,
- Regelkreise zur Überwachung von Fertigungsprozessen und die dazu notwendige Sensorik und Auswertung, sowie
- ausreichend Computerleistung, um Produkte digital zu entwerfen, zu simulieren und die Fertigungsprozesse dazu zu modellieren.

Bauer [33] beschreibt, dass agile Fertigungssysteme eine hohe Volumenflexibilität erlauben, indem standardisierte Bearbeitungszentren, abhängig von der mittelfristigen Auftragslage, in andere Fertigungseinrichtungen integriert oder entfernt werden können. Für Planungssysteme bedeutet es aber auch, dass diese sich möglichst schnell anpassen können müssen. Letztendlich verbessern sich aber durch eine agile Fertigung die Chancen von KMU, die ohne eine starre Struktur, wie sie in größeren Unternehmen üblich ist, gute Aussichten auf die Adaption an ein agiles Fertigungskonzept haben. Besonders die Bildung virtueller Unternehmen, die über ein Fertigungsnetzwerk hinweg Zugang zu deren Dienstleistungen ermöglichen, steigert deren Attraktivität [4].

2.2.2 Rekonfigurierbare Fertigungssysteme

Koren et al. [5, S. 529] definieren ein rekonfigurierbares Fertigungssystem als "ein für schnelle Änderungen der Struktur, Hardware- und Softwarekomponenten ausgelegtes

System, um kurzfristig Produktionskapazitäten und Funktionen innerhalb einer Teilefamilie an plötzliche Marktänderungen oder Vorschriften anzupassen."[3] Die Komponenten eines solchen rekonfigurierbaren Fertigungssystems sind wiederum rekonfigurierbare Maschinen und rekonfigurierbare Steuerungen. Das Ziel eines rekonfigurierbaren Fertigungssystems ist es, die notwendigen Funktionalitäten und Kapazitäten genau dann bereitzustellen, wenn diese notwendig sind [37]. Laut Koren et al. [5] kombinieren diese den hohen Durchsatz an Werkstücken einer flexiblen Transferstraße mit der Flexibilität eines flexiblen Fertigungssystems, während es außerdem schnell und effizient auf Änderungen reagieren kann. Monostori et al. [38] beschreiben, dass ein rekonfigurierbares Fertigungssystem wie ein flexibles Fertigungssystem betrieben werden kann, aber im Gegensatz dazu die schnelle Einführung neuer Technologien und Systemkomponenten unterstützt. ElMaraghy [37] schätzt ein, dass sich die Ansätze von flexiblen Fertigungssystemen und rekonfigurierbaren Fertigungssystemen in Zukunft miteinander vermischen werden. Wiendahl et al. [8] unterscheiden bei diesen Fertigungssystemen den Begriff der allgemeinen Flexibilität mit dem der individuellen Flexibilität, die ein rekonfigurierbares Fertigungsystem ermöglicht. Koren et al. [5] unterscheiden fünf Haupteigenschaften eines rekonfigurierbaren Fertigungssystems:

1. **Modularität** (Modularity) der Komponenten des Fertigungssystems, wie Steuerung, Software, Achsen, Werkzeugen und sonstige Strukturelementen.

2. **Integrierbarkeit** (Integrability) von Komponenten durch Software- und Hardwareschnittstellen.

3. **Anpassungsfähigkeit** (Customization), um Flexibilität für die Herstellung einer bestimmten Teilefamilie zu bieten (customized flexibility) und um die Integration von Steuerungsmodulen mit den notwendigen Steuerungsfunktionen zu bieten (customized control).

4. Schnelle **Konvertierbarkeit** (Convertibility), um Rüstzeiten von Werkzeugen, Teileprogrammen, Spannmitteln und sonstigen Einstellparametern zwischen den Chargen zu minimieren.

5. **Diagnosefähigkeit**, um Qualitätsprobleme aufgrund häufiger Rekonfiguration der Komponenten schnell zu erkennen.

[3] "A Reconfigurable Manufacturing System (RMS) is designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or in regulatory requirements" [5, S. 529].

2.2.3 Holonische Fertigungssysteme

Holonische Fertigungssysteme basieren auf dem 1967 von dem Philosophen Arthur Koestler [39] vorgeschlagenen Begriff *Holon*, bestehend aus dem griechischen Wort *holos*, das *Ganzes* bedeutet, und dem Suffix *on*, das *Teil* bedeutet. Mit diesem Begriff beschreibt er einen identifizierbaren Teil eines Systems, der einerseits aus weiteren Teilen besteht, andererseits Teil eines Ganzen ist [40]. Ein holonisches System oder Holarchie ist eine Hierarchie von Holons mit einer rekursiven Architektur, in dem eine Aufgabe von einem oberen Holon auf mehrere darunterliegende Holons immer tiefer aufgeteilt wird [6]. Holons arbeiten zusammen, indem sie ihre Fähigkeiten und ihr Wissen einbringen, um gemeinsame Ziele des Systems zu erreichen [41].

Van Brussel [42, S. 654-655] definiert ein holonisches Fertigungssystem als "ein Fertigungssystem, das über eine verteilte Steuerung gemäß dem holonischen Systemparadigma verfügt, in dem Komponenten als autonome und kollaborative Agenten (Holons) modelliert sind."[4] Dabei wird die Fertigungssteuerung durch ein verteiltes Multiagentensystem (MAS) koordiniert, das die Interaktionen zwischen den einzelnen Holons steuert. Nach Leitão [41] ist ein holonisches Fertigungssystem eine *Holarchie*, die alle Fertigungselemente wie Maschinen, Produkte, Werkstücke und fahrerlose Transportfahrzeuge umfasst. Darin werden auf Grund der Kernbedürfnisse der Fertigung drei Arten von Holons unterschieden [42]:

- Ein **Ressourcen-Holon** (resource holon) enthält einerseits eine physische Ressource, wie eine Fabrik, eine Maschine oder eine sonstige Komponente, die den anderen Holons Produktionskapazitäten bietet und andererseits einen informationsverarbeitenden Teil, der diese steuert [42].
- Ein **Produkt-Holon** (product holon) enthält das Prozess- und Produktwissen, wie Arbeitspläne, Stücklisten und Verfahren zur Qualitätssicherung [42].
- Ein **Auftrags-Holon** (order holon) ist für die Ausführung eines Produktionsauftrages verantwortlich und besitzt alle dazu alle notwendigen Informationen zum Auftrag und dessen aktuellen Status [42].

Dem Konzept der holonischen Fertigungssysteme folgend, lassen sich besonders komplexe Fertigungssysteme abbilden. Nach Leitão und Restivo [43] werden damit unter anderem Rekonfiguration, Flexibilität, Erweiterbarkeit und Fehlertoleranz von

[4] "A Holonic Manufacturing System (HMS) is a manufacturing system (MS) that is distributively controlled according to the holonic system paradigm. The MS components are modeled as autonomous, collaborative entities (agents), called holons" [42, S. 654-655].

Fertigungssystemen verbessert, weil eine verteilte Steuerung mit autonomen Systemkomponenten eine schnelle und agile Rekonfiguration ermöglicht.

2.2.4 Plug & Produce (P&P)

Arai et al. [6] schlagen mit Plug & Produce (P&P) einen Ansatz vor, um die Inbetriebnahme und Rekonfiguration agiler Systeme zu erleichtern, indem neue Geräte schnell und einfach in die Fertigung integriert und dadurch Rüst- und Rekonfigurationszeiten reduziert werden können. Nach Jasperneite et al. [44] ist P&P eine Merkmalsausprägung zur Kompatibilität eines Fertigungssystems, mit der, äquivalent zur Plug & Play (PnP) Funktion zwischen Computer und Peripherie, neue Module erkannt und automatisch konfiguriert werden können. Dabei bezieht sich P&P hauptsächlich auf mechatronische und informationstechnische Komponenten, weniger auf mechanische Schnittstellen. P&P ist damit eine mögliche Technologie, die agile Fertigungssysteme bei der Inbetriebnahme und Rekonfiguration unterstützt.

Laut Pfrommer et al. [45] müssen Produktionsanlagen durch die Entwicklungen in Richtung kleinerer Bestellmengen bei steigender Anzahl an Varianten immer häufiger umgerüstet und angepasst werden, wobei konventionelle automatisierte Fertigungssysteme oft nur auf wenige Produktvarianten eingeschränkt sind. Umrüsten bei automatisierten Systeme ist oft schwierig und teuer, da Anlagen mit hohem Automatisierungsgrad oft auf einen optimalen Betrieb und weniger auf Austauschbarkeit ausgelegt sind. Außerdem ziehen Rekonfigurationen in automatisierten Anlagen oft weitere Aufwände und Kosten mit sich, wie die erneute Auslegung der Sicherheitsanforderungen nach der Maschinenrichtlinie. Dabei haben nur wenige KMU bereits Automatisierungskompetenzen im eigenen Unternehmen, in den meisten Fällen folgen hohe Kosten durch Systemintegratoren. Schurig et al. [29] kritisieren dabei, dass in vielen Unternehmen Lean-Prinzipien zu konsequent verfolgt werden, um die Produktion auf einen ökonomisch optimalen Betriebspunkt auszurichten, obwohl dieser Betriebspunkt durch die starke Marktvolatilität sehr variiert. Durch P&P sollen letztendlich Kosten bei der Rekonfiguration gespart werden, indem viele der Aufwände, für die ein Systemintegrator benötigt wird, wegfallen, wovon wiederum KMU durch ihre Anpassungsfähigkeit besonders profitierten.

Bührer et al. [46] identifizieren mit der **verhaltensbasierten und physischen Veränderbarkeit** zwei notwendige Stufen der flexiblen Automatisierung. Physische Veränderbarkeit erlaubt es die Anlagentopologie zu verändern, indem Komponenten ausgetauscht, hinzugefügt oder entfernt werden und durch Modellierung und automatische Codegenerierung die Software angepasst wird. Anpassungsfähige Software

auf der Feldebene kann daneben das Verhalten der Anlage entsprechend der Vorgaben eines Manufacturing Execution System (MES) zur Laufzeit ändern. Jasperneite et al. [44] verbinden Rekonfiguration mit der Wandlungsfähigkeit von Fertigungsanlagen, während sich Flexibilität, wie in konventionellen Fertigungssystemen, hier nur auf die Anpassungsfähigkeit der Systeme ohne Änderung der physischen Systemstruktur bezieht.

Bei der Kombination/Verschaltung von Komponenten mit P&P kann zwischen zwei unterschiedlichen Klassen von Anwendungen unterschieden werden: (1) **No-Function-in-Structure** und (2) **Logic-by-Structure**. Diese unterscheiden sich hinsichtlich der Abhängigkeit der Logik der Verschaltung von der Struktur der Verschaltung. Im ersten Fall ist die Logik unabhängig von der Struktur, wie eine Komponente mit einer anderen verschaltet ist. Im zweiten Fall ist die Logik auch von der Struktur der Verschaltung der beiden Komponenten, also der Anlagentopologie, abhängig, die daher ebenfalls beschrieben werden muss [44], [47]. Ein Kernthema, das aus der Verschaltung mehrerer Komponenten entsteht ist die möglichst automatisierte Generierung der Integrationssoftware für spezifische Kombinationen. Diese erlaubt die Komposition der Fähigkeiten der einzelnen Komponenten. Nach Jasperneite et al. [44] kann bei bestehenden P&P Ansätzen derzeit zwischen drei Gruppen unterschieden werden:

1. **Modellbasierte Softwareentwicklung** erlaubt es das Wissen von Experten durch Modellierung zu formalisieren und daraus meist automatisiert Code für die rekonfigurierte Anlage zu generieren. [44].

2. **Agentenbasierte Ansätze** ermöglichen nach dem Prinzip der Selbstorganisation einzelner Agenten eine automatische und verteilte Ausführung einer implizit vorhandenen Integrationssoftware [44]. Viele dieser MAS-basierten Ansätze [6], [41], [48] treten im Zusammenhang mit holonischen Fertigungssystemen auf.

3. **Softwaregenerierung** ist ein weiterer vielversprechender Ansatz, bei dem nur noch eine Zielvorgabe modelliert wird aus der, mit der vorhandenen Produkt- und Prozessbeschreibung, automatisiert die Integrationssoftware generiert wird [44].

Naumann et al. [49] liefern die notwendigen Methoden für Kombination von Geräten in einem Ansatz mit modellbasierter Softwareentwicklung und möglicher Softwaregenerierung: (1) Eine Evaluierung möglicher Kombinationen von Geräten. (2) Die Generierung neuer Gerätebeschreibungen aus den Beschreibungen der Einzelkomponenten. (3) Die Identifikation der ausführbaren Prozesse in der Kombination der Geräte. (4) Die Generierung von Steuerungssequenzen für alle Prozesse.

2.2.5 Cyber-physische Produktionssysteme (CPPS)

CPPS sind eine Ausprägung von CPS im Bereich der Produktion, die nach Monostori [7] aus autonomen und kooperativen Elementen bestehen, die über alle Produktionsebenen hinweg miteinander kontextbezogen verknüpft sind. Dadurch werden besonders die höheren Hierarchieebenen der traditionellen Automatisierungspyramide durch eine für CPPS charakteristische **dezentrale Funktionsweise** ersetzt. CPPS zeichnen sich weiterhin aus durch: (1) **Intelligenz und Autonomie**, (2) **Konnektivität** mit Mensch und Maschine und (3) **Reaktionsfähigkeit** auf Veränderungen [38]. Laut Siepmann [50] können CPPS als ein Verbund von CPS die Produktion dezentral und kontextadaptiv auch über Unternehmensgrenzen hinweg steuern.

CPS bilden dabei die Grundlage für CPPS, indem sie die Verbindung von physikalischer und informationstechnischer Welt schaffen. Nach der acatech Studie: Integrierte Forschungsagenda Cyber-Physical Systems [51, p. 244] umfassen CPS "eingebettete Systeme, Logistik-, Koordinations- und Managementprozesse sowie Internetdienste, die mittels Sensoren unmittelbar physikalische Daten erfassen und mittels Aktoren auf physikalische Vorgänge einwirken, mittels digitaler Netze untereinander verbunden sind, weltweit verfügbare Daten und Dienste nutzen und über multimodale Mensch-Maschine- Schnittstellen verfügen."

Siepmann und Graef [50] beschreiben die Entwicklung zu I4.0 als dreistufiges Modell, in dem die höchste Stufe I4.0 als Zukunftsvision für Unternehmensstrategien und neue Geschäftsmodelle auf den beiden unteren Stufen CPS und CPPS aufbaut. Die Grundlagen eines CPS bilden darin die Bausteine **Ubiquitous Computing**, also die generelle Zunahme von Geräten mit Rechenleistung zur Datenverarbeitung in allen Lebensbereichen, das **Internet of Things and Services (IoTS)** und die darüber ermöglichte Kommunikationsfähigkeit zwischen diesen Geräten und mittels **Cloud Computing** auch die Möglichkeit, Rechenleistung anderer Ressourcen zeitweise zu nutzen. In einem CPPS ist es laut den Autoren notwendig, dass die Kommunikation zwischen Maschinen, ebenso wie die Kommunikation zwischen Mensch und Maschine, mit geeigneten Schnittstellen ermöglicht wird. Dazu bedarf es besonders bei der Maschine-zu-Maschine Kommunikation der Standardisierung.

2.3 Agil-rekonfigurierbare Fertigungssysteme

Mit den Herausforderungen der individuellen Produktion und der zunehmenden Digitalisierung unterliegen auch konventionelle Fertigungssysteme einem Wandel: angestoßen

von der Vision der agilen Fertigung zu Beginn des 21. Jahrhunderts, über holonische und rekonfigurierbare Fertigungssysteme, hin zu CPPS. Dabei sind diese Fertigungskonzepte oft nur schwierig greifbar, da agile Fertigung eher eine Unternehmensstrategie beschreibt, als ein Fertigungssystem und CPPS eher einen Ausblick auf künftige Fertigungssysteme bietet, als Lösungsansätze zu generieren. Diese gilt es aktuell mit I4.0 noch zu finden und zu definieren. Holonische Fertigungssysteme dagegen haben mit der darunterliegenden Technologie der MAS bereits einen gewissen technologischen Reifegrad erreicht, sind aber aufgrund ihrer Komplexität noch nicht in der Industrie angekommen, die eher auf herkömmliche zentrale Steuerungsansätze hinarbeitet. Rekonfigurierbare Fertigungssysteme hatten dagegen ihre ersten Ausprägungen bereits im ersten Jahrzehnt des 21. Jahrhunderts und haben sich zumindest durch immer leichter austauschbare Komponenten von Maschinen und Fertigungszellen in der Industrie niedergeschlagen. Zuletzt bietet auch der P&P Ansatz keine standardisierte Lösung, die für einen industriell tauglichen Einsatz notwendig wäre.

Jedoch sind einige der Prinzipien, die von diesen Ansätzen geliefert werden, sehr gut für die zukünftigen Entwicklungen im Rahmen von I4.0 geeignet, denn alle Ansätze arbeiten grundsätzlich auf ein ähnliches Ziel hin, und zwar den Weg von der Massenproduktion hin zur individuellen Produktion durch modulare, schnell reaktionsfähige und durch Informationstechnik unterstützte Organisationen und Fertigungssysteme. Die daraus entstehenden **Fertigungssysteme sind agil-rekonfigurierbar** und können sich durch die zumindest **teil-automatisierte Komposition ihrer Fähigkeiten** an die aktuelle Konfiguration anpassen. Als komplexe Systeme sind diese nach den P&P Prinzipien oft **Logic-by-Structure Systeme**, in denen die Konfiguration und automatische Softwaregenerierung von der Struktur der Verschaltung der Module abhängt. Die Softwarekomposition kann entweder durch eine verteilte Steuerung nach den MAS Prinzipien oder durch einen zentralen Ansatz erfolgen, wobei für die Industrie standardisierbare Lösungen noch gefunden werden müssen.

Kapitel 3

Die Grenzen der semantischen Interoperabilität

Nach dem European Research Cluster on the Internet of Things (IERC) und Serrano et al. [52] werden durch die Entwicklungen im Bereich der Informations- und Kommunikationstechnologien (IKT) immer mehr kommunikationsfähige Geräte generiert. Mit der stark zunehmenden Digitalisierung in der Industrie im Laufe der letzten beiden Jahrzehnte wurde damit die Interoperabilität zwischen Geräten verschiedener Herstellerfirmen zu einem zunehmend wichtigeren Thema. Nach der Definition des Institute of Electrical and Electronics Engineers (IEEE) [24, S. 114] ist Interoperabilität dabei "die Fähigkeit zwischen Geräten Informationen auszutauschen und die ausgetauschten Informationen zu nutzen."[5] Die Plattform I4.0 [53] dagegen definiert Interoperabilität als **"Fähigkeit zur aktiven, zweckgebundenen Zusammenarbeit von verschiedenen Komponenten, Systemen, Techniken oder Organisationen"**.

3.1 Sichtweisen der Interoperabilität

In den letzten zwei Jahrzehnten haben sich viele Forschende mit der Interoperabilität zwischen unterschiedlichen Geräten und softwaretechnischen Anwendungen beschäftigt. Ein Großteil der Modelle zur Interoperabilität setzen als Grundlage entweder auf die Semiotik (Theorie der Zeichen) des 20. Jahrhunderts der Philosophen Charles San-

[5] "[...] the ability of two or more systems or components to exchange data and use information" [24, S. 114].

ders Peirce und Charles William Morris, oder auf die oft damit in Beziehung gesetzte Wissenspyramide.

3.1.1 Wissenspyramide

Zur Wissenspyramide gibt es viele, oft sehr ähnliche Modelle. Das derzeit im Rahmen der Digitalisierung und I4.0 unter anderem meistzitierte ist die Wissenspyramide (Abbildung 3.1) nach Fuchs-Kittowski [54], die in Anlehnung an das *Data-Information-Knowledge Model* von Aamodt und Nygård [55] den Zusammenhang und den Übergang von Zeichen zu Daten, Informationen und Wissen zeigt und damit auch den Übergang von der menschlichen (semantischen) zur maschinellen (syntaktischen) Informationsverarbeitung. Fuchs-Kittowski [54] sieht dabei Information als Kombination von Struktur, Bedeutung und Wirkung, oder aus Sicht der Semiotik, wie sie durch den amerikanischen Philosophen Charles William Morris [56] geprägt wurde, als Kombination von Syntax, Semantik und Pragmatik.

Damit kann die Wissenspyramide neben der menschlichen Kommunikation, auch zur Veranschaulichung der Mensch-Maschine-Kommunikation und der Maschine-Maschine-Kommunikation herangezogen werden (Abbildung 3.2). Ein Sender generiert dabei zielorientierte Informationen aus seinem Wissen, welche abstrahiert zu Daten und letztendlich zu übertragbaren Zeichen (bzw. Signalen) encodiert werden, die vom

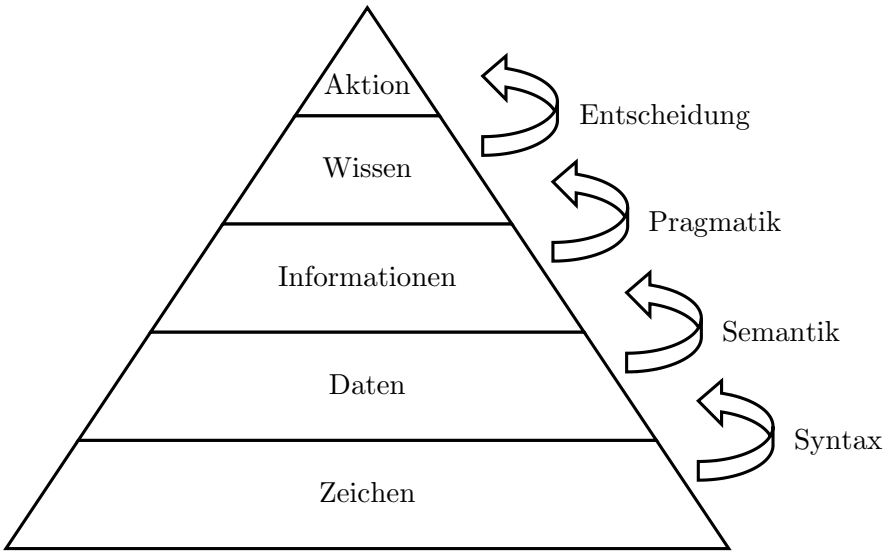


Abbildung 3.1: Wissenspyramide nach Fuchs-Kittowski [54].

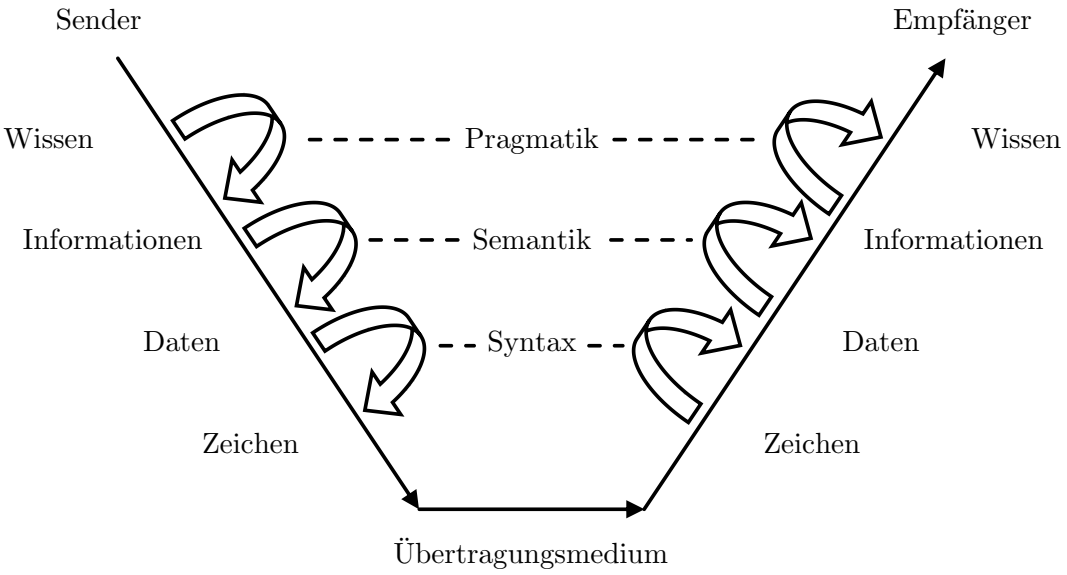


Abbildung 3.2: Sender-Empfänger-Modell, in Anlehnung an das erweiterte Kanalmodell nach Fuchs-Kittowski [54].

Empfänger über das jeweilige Kommunikationsmedium empfangen werden. Der Empfänger verwendet wiederum den entsprechenden Syntax, um diese Zeichen zu Daten zu decodieren. Diese Daten werden zu Informationen, wenn sie mittels Semantik, also der Bedeutung der Daten, interpretiert werden können. Dabei ist wichtig, dass das Verständnis der Daten beim Sender und beim Empfänger eindeutig ist, da sonst Fehlschlüsse gezogen werden können.

Folglich ist neben der Standardisierung von Übertragungsmedium und Syntax auch die Frage nach der Standardisierung der Bedeutung der kommunizierten Daten relevant. In den höheren Ebenen der Wissenspyramide wird jedoch Wissen benötigt, um als Grundlage für eine Entscheidungsfindung zu dienen, um letztendlich eine Aktion herbeizuführen. Dieses Wissen ergibt sich nach Fuchs-Kittowski [54] aus begründeten, miteinander in Beziehung gesetzten Informationen, wie Ursache-Wirkungs-Beziehungen. Neben der Semantik und der Frage nach der Bedeutung von Daten, stellt sich also mit der Pragmatik die Frage nach dem Nutzen und Effekt ausgetauschter Daten.

3.1.2 Levels of Conceptual Interoperability Model (LCIM)

Ein auf der Semiotik aufbauendes Modell zur Interoperabilität liefert Tolk [57] im Jahr 2003 mit dem Levels of Conceptual Interoperability Model (LCIM) (Abbildung 3.3), das später durch Turnitsa [58] erweitert wurde und Interoperabilität in sechs aufeinander aufbauenden Schichten bis hin zur konzeptuellen Interoperabilität beschreibt [59]:

- **Technische Interoperabilität (technical)** beschreibt die Kommunikation zwischen Systemen, mit der Daten über eine Infrastruktur und ein bestimmtes Protokoll ausgetauscht werden.
- **Syntaktische Interoperabilität (syntactic)** beschreibt die einheitliche Struktur, in der diese ausgetauschten Daten organisiert sind.
- **Semantische Interoperabilität (semantic)** bedeutet, dass ein einheitliches Referenzmodell verwendet wird, in dem die Bedeutung der Daten definiert wird und das die ausgetauschten Daten somit eindeutig beschreibt.
- **Pragmatische Interoperabilität (pragmatic)** bedeutet, dass interagierende Systeme gegenseitig ihre Fähigkeiten und Methoden und damit die Verwendung der übertragenen Daten kennen und in welchem Kontext diese angewendet werden.
- **Dynamische Interoperabilität (dynamic)** beschreibt die Veränderung des Zustands eines Systems über die Zeit und die damit einhergehende Veränderung von Annahmen und Bedingungen. Folglich ist eine Veränderung des Zustands auch eine Veränderung des Kontexts in dem ein System verwendet wird, wodurch abhängig vom Zustand andere Regeln gültig sind, die das System beherrschen muss. Desweiteren kann sich ebenfalls der Effekt einer gewissen Aktion abhängig vom Zustand des Gesamtsystems ändern.
- **Konzeptuelle Interoperabilität (conceptual)** beschreibt die Übereinstimmung der Annahmen und Bedingungen einer zweckgebundenen Abstraktion der Realität mit dem System. Dieses Modell muss ausreichend spezifiziert sein, damit alle notwendigen Bedingungen darin vorhanden sind, muss aber gleichzeitig unabhängig von der tatsächlich implementierten Lösung sein. Wang et al. [60] beschreiben so ein konzeptuelles Modell als eine abstrahierte und vereinfachte Repräsentation von Systemen, die einem bestimmten Zweck dient.

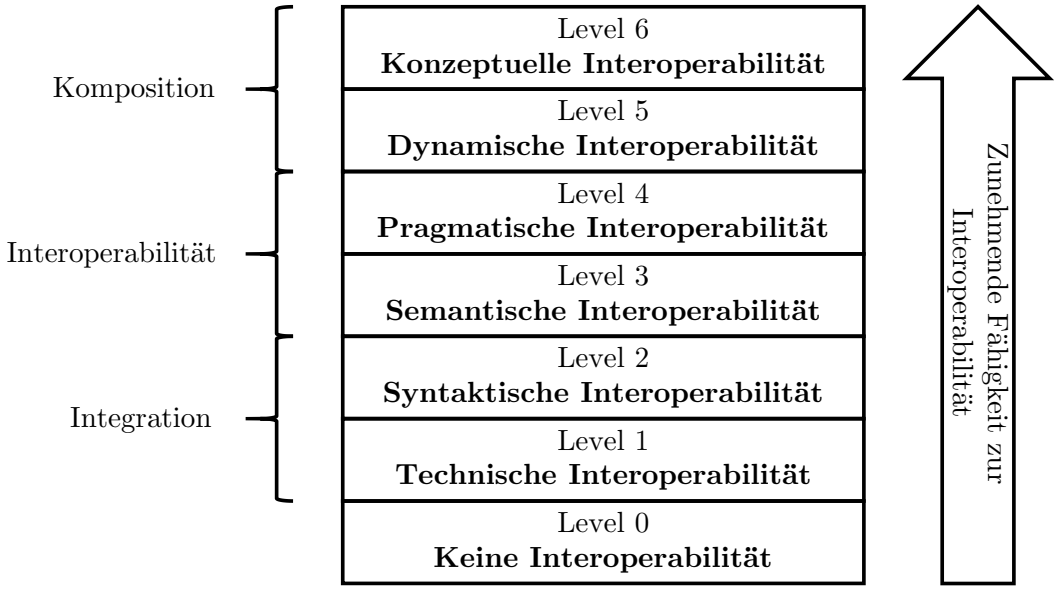


Abbildung 3.3: Schichten der Interoperabilität im Levels of Conceptual Interoperability Model (LCIM) [57], [58].

3.1.3 Dimensionen der Interoperabilität nach ETSI und IERC

Serrano et al. [52] orientieren sich im Bericht des IERC zu IoT an den Dimensionen der Interoperabilität nach Van Der Veer und Wiles vom European Telecommunications Standards Institute (ETSI) [61], die neben **technischer, syntaktischer und semantischer Interoperabilität** auch die **organisatorische Interoperabilität** sehen. Zusätzlich nennen Serrano et al. [52] mit der **dynamischen und statischen Interoperabilität** zwei weitere Trends zur Interoperabilität im Bereich IKT, bei denen im Gegensatz zur vollständigen Standardisierung von IoT Geräten akzeptiert wird, dass es Unterschiede zwischen Geräten gibt und sich diese zur Laufzeit anpassen müssen. Dynamische Interoperabilität ist laut IERC mit der steigenden Komplexität und Heterogenität im Bereich IoT ein wichtiges Forschungsgebiet, damit unterschiedliche Geräte dynamisch miteinander interagieren können. Demnach müssen Geräte zur Interaktion eine gemeinsame Menge an Services besitzen und diese mit allen notwendigen Bedingungen beschreiben.

- **Organisatorische Interoperabilität (organisational)** ist die Fähigkeit Informationen durch verschiedenste Informationssysteme und Infrastrukturen zu verteilen, die möglicherweise auch geografisch und kulturell unterschiedlich sind

[61].

- **Statische Interoperabilität (static)** beschäftigt sich mit der statischen Konformitätskontrolle aller statischen und dynamischen Voraussetzungen und überprüft damit, ob eine Anwendung mit mehreren verteilten Geräten und Services alle Anforderungen erfüllt [52].

3.1.4 Semiotic Interoperability Framework

Li et al. [62], [63] liefern mit ihrem Modell zur semiotischen Interoperabilität (Semiotic Interoperability Framework) ebenfalls eine neue Sichtweise auf das Thema Interoperabilität, bei welcher der soziale und der technische Aspekt von Organisationen und Informationssystemen im Vordergrund steht. Die Autoren bedienen sich dabei dem *Semiotic Framework* von Stamper [64] und beschreiben, dass durch die semiotische Interoperabilität Informationssysteme zusammenarbeiten können durch Kommunikation mit Einsicht in **physische Eigenschaften**, die **Übertragungsstruktur von Zeichen**, sowie die **Kommunikation von Bedeutung, Zweck und sozialen Konsequenzen** von Informationen [62]. Die Komponenten zur semiotischen Interoperabilität sind in aufsteigender Reihenfolge [62]:

- **Physische Interoperabilität (physical)** stellt die Kommunikation zwischen zwei Geräten dar, bei der Signale von einem Sender an einen Empfänger übertragen werden.
- **Empirische Interoperabilität (empiric)** ist aufbauend auf der physikalischen Interoperabilität die korrekte Enkodierung und Dekodierung von Signalen über geeignete Kanäle und Protokolle. Physische und empirische Interoperabilität ergeben folglich in der Kombination die vorher beschriebene **technische Interoperabilität**.
- **Syntaktische Interoperabilität (sytactic)** entsprechend der vorhergehenden Definition.
- **Semantische Interoperabilität (semantic)** entsprechend der vorhergehenden Definition.
- **Pragmatische Interoperabilität (pragmatic)** stellt sicher, dass die Prozesse von einzelnen Geräten, die sich auf deren eigenen Kontext als Einzelkomponente beziehen, ebenso mit anderen Geräten aggregiert werden können, um ein übergeordnetes Ziel zu erreichen. Damit schließen die Autoren die Fähigkeit zur

Komposition in die pragmatische Interoperabilität mit ein, die im LCIM eher in der Ebene der dynamischen Interoperabilität zu finden ist.

- **Soziale Interoperabilität (social)** stellt sicher, dass das resultierende System kohärent mit den sozialen Normen und Zielen der Organisation ist. Dabei soll der Zweck, den ein Sender erreichen wollte, beim Empfänger auch zu einer gewünschten sozialen Konsequenz führen, die abhängig von der Wahrnehmung der jeweiligen Organisation ist.

3.1.5 Interoperabilität nach ISO/IEC 21823-1

Letztendlich wurden in der 2020 veröffentlichten Roadmap zu I4.0 durch das Deutsche Institut für Normung (DIN) und die Deutsche Kommission Elektrotechnik (DKE) [23] als Leitbild für das folgende Jahrzehnt drei Strategiebereiche für digitale Ökosysteme genannt: **Autonomie, Interoperabilität und Nachhaltigkeit**. Interoperabilität erlaubt es dabei in diesen Ökosystemen auch über Unternehmensgrenzen hinaus mit Unternehmen und Einzelpersonen in Wertschöpfungsnetzwerken zu interagieren. Dabei wird hier ebenfalls auf die Wissenspyramide (Abbildung 3.1), mit der Kombination aus Syntax, Semantik und Pragmatik als Grundlage für die Interoperabilität verwiesen. Desweiteren wird auch auf die Standarisierung der Interoperabilität nach der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC) in ISO/IEC 21823-1 [65] verwiesen:

- **Transportinteroperabilität (transport)** verwendet eine etablierte Kommunikationsinfrastruktur zwischen den teilnehmenden Systemen. Sie kann als Standardisierung des Übertragungsmediums, wie im Sender-Empfänger Modell (Abbildung 3.2) gesehen werden und ist damit mehr oder weniger mit der **technischen Interoperabilität** gleichzusetzen.
- **Syntaktische Interoperabilität (syntactic)** verwendet Formate für die ausgetauschten Daten, die von den teilnehmenden Systemen verstanden werden können und liefert damit die Struktur.
- **Semantische Interoperabilität (semantic)** bildet die Bedeutung der Daten ab. Wichtig ist, das hier auch auf das Anwendungsgebiet und somit den Kontext der Daten hingewiesen wird, damit Daten von den teilnehmenden Systemen verstanden werden.
- **Richtlinieninteroperabilität (policy)** stellt sicher, dass die teilnehmenden Systeme vorgegebenen rechtlichen und organisatorischen Richtlinien entsprechen.

Dabei handelt es sich zum Beispiel um Rechtslage und Datensicherheit von Geräten in unterschiedlichen Gerichtsbarkeiten.

- **Verhaltensbasierte Interoperabilität (behavioral)** stellt sicher, dass das Ergebnis auch der Erwartung entspricht. Dies spiegelt ebenfalls die Sichtweise in der Definition der pragmatischen Interoperabilität durch Asuncion und Sinderen [66], [67] wieder.

3.1.6 Interoperabilität im Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)

Das Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) (DIN SPEC 91345 [68]) ist eine im Jahr 2015 vom Verein Deutscher Ingenieure (VDI) / Verband der Elektrotechnik, Elektronik und Informationstechnik (VDE) - Gesellschaft Mess- und Automatisierungstechnik und dem Zentralverband Elektrotechnik- und Elektronikindustrie (ZVEI) [69] vorgestellte dreidimensionale Referenzarchitektur für I4.0 (Abbildung 3.4). Laut Adolphs und Epple [69] soll es die Vernetzung von Produktionsmitteln über die vertikale Integration in Unternehmen, also über die Hierarchielevel (Hierarchy Levels), zusammen mit der horizontalen Integration, also dem Engineering und der Wertschöpfungskette über den Lebenszyklus (Life Cycle Value Stream), in einem Modell abbilden. Als dritte Achse beschreibt das RAMI4.0 einen hierarchisch strukturierten Ansatz für die IT-Repräsentanz von I4.0-Komponenten über sechs Schichten.

Laut RAMI4.0 [69] kann eine Komponente einen allgemeinen physischen Gegenstand oder ein informationstechnisches Objekt darstellen, eine I4.0-Komponente ist jedoch eine spezielle Komponente, die durch eine Verwaltungsschale zur I4.0-Komponente wird, einer Spezialisierung eines CPS, das die folgenden Merkmale und Anforderungen besitzt: eindeutige Identifizierbarkeit, I4.0-konforme Kommunikation mit I4.0-konformer Semantik und I4.0-konformen Diensten und Zuständen, eine virtuelle Beschreibung, einschließlich des dynamischen Verhaltens, Security und Safety, Quality of Services und Schachtelbarkeit von Komponenten. Die Verwaltungsschale ist dabei das digitale Abbild eines Gegenstandes oder Assets, das diesen über seinen Lebenszyklus hinweg beschreibt [71]. Laut Bedenbender et al. [72] muss die Verwaltungsschale dazu eine minimale, aber hinreichende Beschreibung von I4.0-Komponenten bereitstellen, mit der bereits vorhandene **Standards in Teilmodellen** klassifiziert werden können. Solche Teilmodelle und deren spezifische Eigenschaften sollen für jede technische Domäne, wie *Bohren* oder *Assemblieren*, einmal standardisiert werden und mit anderen komplementären Teilmodellen ergänzt werden können, damit diese mehrere Aspekte einer I4.0-Komponente beschreiben können. Laut Bader et al. [73] ist es dabei aber auch

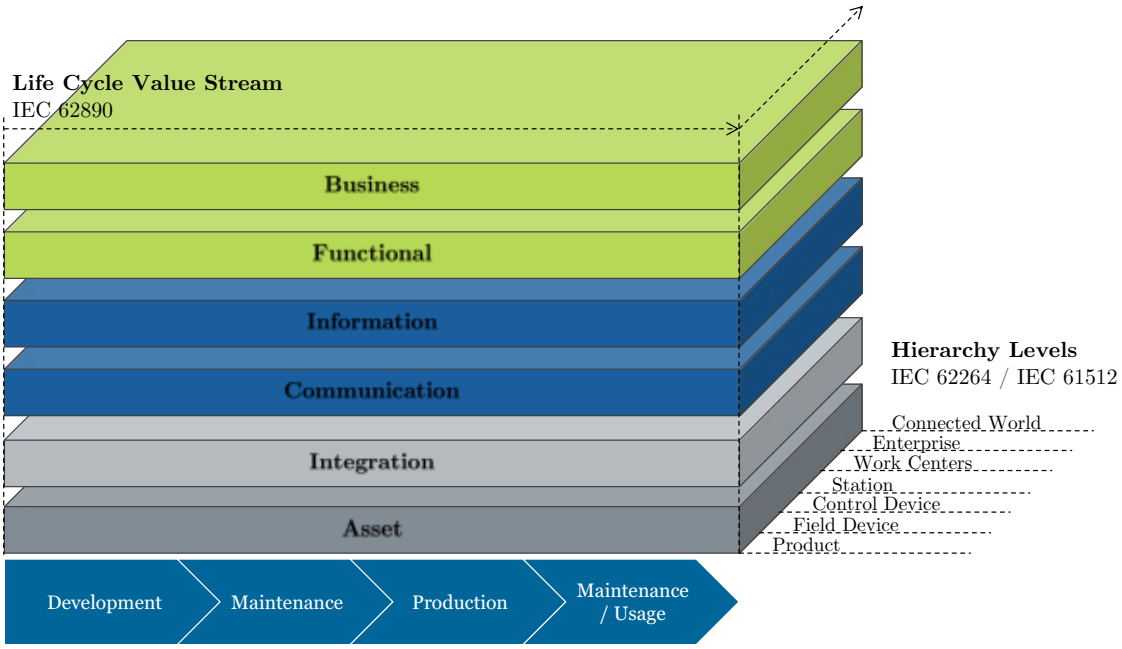


Abbildung 3.4: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) [69], [70].

notwendig für I4.0, dass möglichst viele freie und auch proprietäre Teilmodelle erstellt werden können. Dazu werden diese standardisierten Teilmodelle im Metamodell zur Verwaltungsschale (Version 2.0.1, Abschnitt 4.4.5 und 4.4.8) [73] durch individuelle proprietäre oder anwendungsspezifische Identifikatoren erweitert, die aber von anderen Systemen ignoriert werden sollen, wenn diese nicht bekannt sind. I4.0-Komponenten können in die folgenden Ebenen von RAMI4.0 in aufsteigender Reihenfolge eingeordnet werden:

1. Die **Gegenstandsschicht (Asset Layer)**, über die physikalische, reale Gegenstände, aber auch Menschen [69] mit den I4.0-Komponenten eingebunden sind.
2. Die **Integrationsschicht (Integration Layer)**, in der die Schnittstelle zwischen dem Gegenstand oder Menschen und der digitalen Datenverarbeitung hergestellt wird [69], also die Ebene, ab der die Verwaltungsschale ansetzt.
3. Die **Kommunikationsschicht (Communication Layer)**, in der die bidirektionale Kommunikation mit der Integrationsschicht über ein einheitliches Datenformat ermöglicht wird, so dass die darüberliegende Informationsschicht darauf ansetzen kann [69].
4. Die **Informationsschicht (Information Layer)**, die sich mit der Datenintegrität, dem Speichern, Bereitstellen, Transformieren und Aggregieren von Daten

und der Integration von Datenmodellen beschäftigt, sowie auf Regeln basierende Ereignisse für die Funktionsschicht erzeugt [69].

5. Die **Funktionsschicht (Functional Layer)**, in der Regeln und Funktionen erzeugt, beschrieben, Dienste modelliert und dann dort oder in den darunterliegenden Schichten ausgeführt werden [69].

6. Die **Geschäftsschicht (Business Layer)** ist eine abstrakte Schicht, mit der die Funktionalität der I4.0-Komponente in die Wertschöpfungskette und den Gesamtprozess integriert wird. Laut RAMI4.0 [69] zählen zu den Aufgaben der Schicht die Abbildung der Geschäftsmodelle, sowie rechtliche Rahmenbedingungen und die Orchestrierung der Dienste der Funktionsschicht.

RAMI4.0 selbst ist im Gegensatz zu den vorherigen Modellen zu Interoperabilität eher an einer Architektur orientiert, in der verschiedene Technologien angesiedelt werden können und die auch weitere Aspekte, wie die Datenhaltung, in Betracht zieht, die in anderen Modellen eher abstrakt gehalten werden. Auch wenn sich sich die Bezeichnungen der Schichten von den bisherigen Interoperabilitätsmodellen unterscheiden, gibt es doch einige Ähnlichkeiten, so dass die Architektur mit den bisherigen Modellen verglichen werden kann: die unterste Gegenstandsebene bezieht sich z. B. auf den Gegenstand selbst, der noch keine zusätzlichen Eigenschaften zur Interoperabilität besitzt. Diese Ebene kann auch mit der untersten Ebene im LCIM (Level 0 - keine Interoperabilität) verglichen werden. Die Ebenen Integration und Kommunikation sind mit den Ebenen zur technischen und syntaktischen Interoperabilität vergleichbar (Abbildung 3.3), die die Integration des Gegenstandes bis zu dessen Kommunikationsfähigkeit mit dem System beschreiben. Ein Teilaspekt der Informationsschicht im RAMI4.0 wird durch die semantische Interoperabilität abgedeckt, wenngleich darin zusätzlich auch weitere Funktionen zur Datenhaltung integriert sind. Funktionsschicht und pragmatische Interoperabilität sind sich ebenfalls ähnlich, wenn man die Definition im LCIM betrachtet. Die Definition im *Semiotic Interoperability Framework* inkludiert aber zusätzlich die Komposition von Diensten, während RAMI4.0 eher die Modellierung und Ausführung von Diensten beschreibt. Die Ebene der Geschäftsschicht kann teilweise mit der Ebene der konzeptuellen Interoperabilität im LCIM verglichen werden, da sie die Integration der Funktionsebene in Wertschöpfungsprozesse beschreibt. Außerdem werden Regeln zur Zielerfüllung im Sinne der sozialen Interoperabilität (Semiotic Interoperability Framework) und rechtliche Rahmenbedingungen im Sinne der Richtlinieninteroperabilität nach ISO/IEC 21823-1 definiert.

Um nun eine Interaktion zwischen I4.0-Komponenten zu ermöglichen, definiert die VDI/VDE 2193 [74], [75], laut der die Kompatibilität von Verwaltungsschalen auf der

Interoperabilität ihrer Teilemodelle basiert, eine Sprache zur Kommunikation von I4.0-Komponenten, die auf der Kombination von Vokabular, Struktur und Interaktionsprotokollen aufbaut. Die Richtlinie klassifiziert dazu Interaktionen nach Reich [76] nach ihrem **Informationsfluss** (in uni- und bidirektionale), ihrem **Zustand** (in zustandsbehaftete und zustandslose), nach ihrem **Determinismus** (in deterministische und nicht deterministische), ihrer **Synchronizität** (in synchrone und asynchrone) und zusätzlich nach ihrer **Symmetrie** (in symmetrische und asymmetrische Interaktionen). Das Konzept zur Interaktion (Abbildung 3.5) baut dabei auf dem Austausch von Nachrichten anhand eines Interationsprotokolls auf, in denen Zustandsübergänge augetauscht werden und damit gegenseitig Verhaltensänderungen bewirkt werden können.

Aufbauend auf dem Interaktionsmodell von I4.0-Komponenten in VDI/VDE 2193 [74], [75] (Abbildung 3.5) liefern Neubacher et al. [77] mit dem **Referenzmodell semantischer Interoperabilität** einen "Vorschlag zur systematischen Klassifikation von Interaktionen in Industrie 4.0 Systemen". Im Unterschied zu den bisherigen Interoperabilitätsmodellen handelt es sich hierbei aber weitestgehend um ein Modell zu Klassifikation von Technologien für Interaktionen entsprechend der Klassifikation in VDI/VDE 2193-1 [74]. Die Autoren stellen fest, dass sich dabei durch horizontale Interaktionen zwischen unterschiedlichen Systemen immer größere und dichtere Interaktionsnetzwerke bilden. Folglich nimmt die Komplexität in solchen Systemen immer

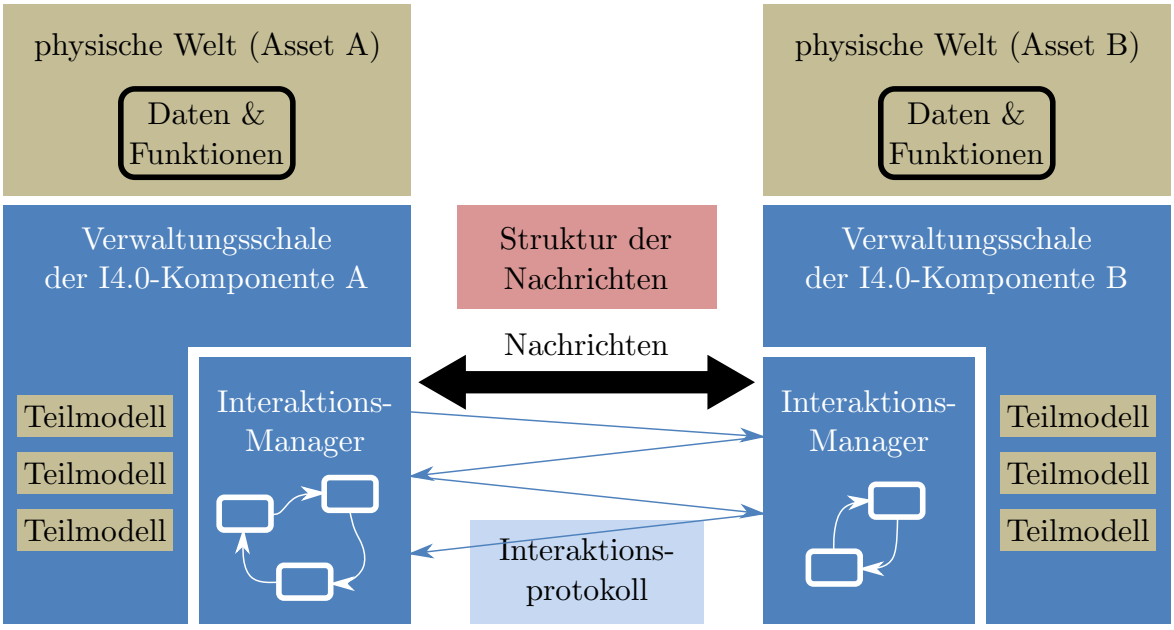


Abbildung 3.5: Interaktionsmodell zwischen I4.0-Komponenten nach VDI/VDE 2193-1 [74].

weiter zu. Außerdem stellen sie fest, dass eine Schnittstelle zwischen zwei Systemen das Transformationsverhalten beschreiben muss, damit weitere Aussagen über die Interoperabilität getroffen werden können. Dabei definieren sie den Begriff der semantischen Interoperabilität unter der Berücksichtigung des Transformationsverhaltens der Interaktion im Kontext des Systemzustandes. Diese Sichtweise zur Interaktionssemantik wurde bereits durch Diedrich et al. [78] in Bezug auf informale Semantik und die Bedeutungszuweisung im Zusammenhang mit Transitionen in Automaten auf Empfängerseite hervorgehoben. Folglich schließen Neubacher et al. [77] die Pragmatik in ihre Definition der semantischen Interoperabilität mit ein. Die Pragmatik und Semantik müss sicherlich im Sinne der tatsächlichen Bedeutung einer Interaktion im Kontext des Systems miteinander verknüpft werden. Aus Sicht der Semiotik ist diese Definition des Begriffs der *semantischen Interoperabilität* aber eher missverständlich, sind doch Syntax, Semantik und Pragmatik bereits für diese einzelnen ebenbürtigen Aspekte definiert. Ein passenderer Titel für diese Sichtweise wäre *semiotische Interoperabilität*.

3.1.7 Interoperabilität nach Industrial Internet Reference Architecture (IIRA)

Als Gegenstück zum RAMI4.0, das hauptsächlich für die Produktionsindustrie erstellt wurde, gibt es aus dem IoT Bereich mit der Industrial Internet Reference Architecture (IIRA) [79] ein weiteres Modell, das vom Industrial Internet Consortium (IIC) im Kontext von IIoT für alle Industriebereiche entwickelt wurde [25]. Mit dieser standardbasierten und offenen Architektur soll ein einheitlicher und branchenübergreifender Ansatz geschaffen werden, der die Interoperabilität von Systemen in verschiedenen Branchen verbessern soll, indem die Architektur auf einem hohen Abstraktionslevel definiert wird. Dieses abstrahiert systemspezifische und proprietäre Eigenschaften und Muster (*System Characteristics* in Abbildung 3.6) und erhält damit einen breiten Anwendungsbereich [79]. Neben diesen systemspezifischen Eigenschaften definiert das IIRA [79] die funktionalen Domänen (*Functional Domains*), die die allgemeinen Systemfunktionen und -fähigkeiten, sowie Querschnittsfunktionen (*Crosscutting Functions*) von IIoT-Anwendungen beschreiben, die über viele der funktionalen Systemkomponenten hinweg notwendig sind. Die funktionalen Domänen des IIRA sind mit den funktionalen Ebenen des RAMI4.0 vergleichbar. Laut IIC und Plattform-I4.0 [80] ist dabei die Ebene der **physischen Systeme (Physical Systems)** in der IIRA keine funktionale Domäne und wurde damit nicht formal definiert, kann aber im Vergleich zum RAMI4.0 mit der Gegenstandsschicht (*Asset Layer*) verglichen werden, auch wenn ein Asset in RAMI4.0 fast alles beschreibt, das am Wertschöpfungsprozess teilnimmt und sich in der IIRA nur auf ein überwachtes und gesteuertes physisches Objekt bezieht. Ein weiterer Unterschied ist der Kommunikationsschicht, die in der IIRA (*Connectivity and*

Distributed Data) eine Querschnittsfunktion darstellt.

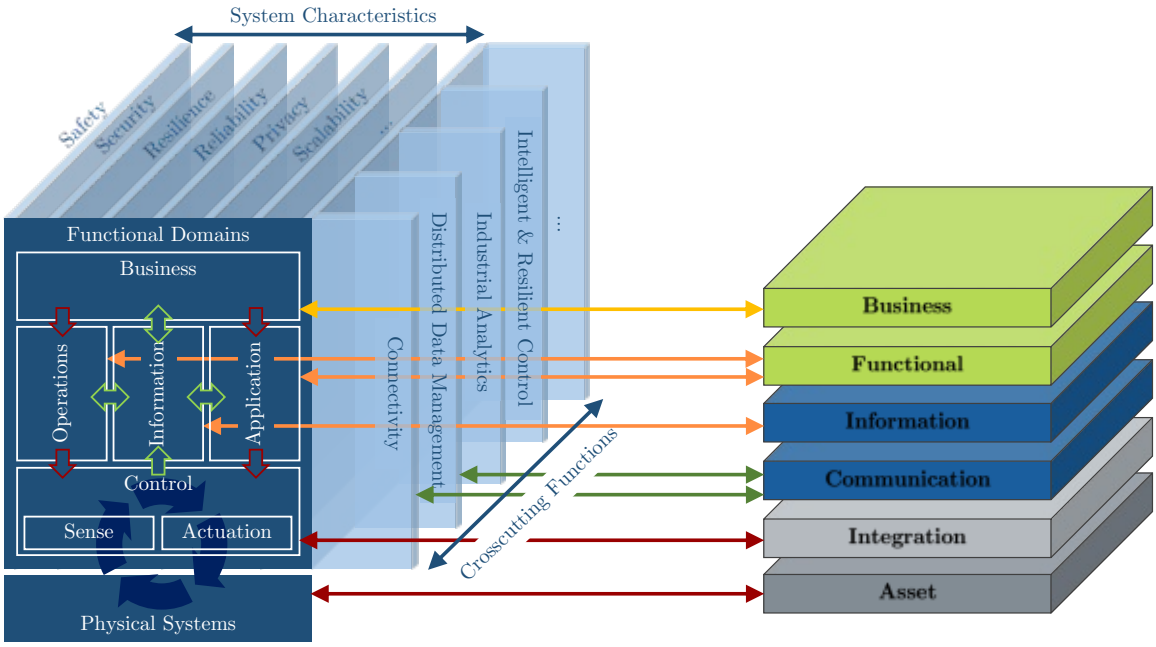


Abbildung 3.6: Industrial Internet Reference Architecture (IIRA) vs. Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) nach Industrial Internet Consortium (IIC) und Plattform-Industrie 4.0 (I4.0) [80].

1. Der **Steuerungsbereich (Control Domain)** stellt die Schnittstelle mit den physischen Systemen dar, in dem Steuerungs- und Regelungssysteme, oft in Echtzeit und mit hoher Auflösung, mit Sensoren und Aktoren interagieren [79]. Im RAMI4.0 stellt analog die Integrationsschicht die Schnittstelle zum Gegenstand und der Prozesssteuerung dar, mit der Ausnahme, wenn es sich dabei um eine I4.0-konforme Schnittstelle handelt. Diese besitzt nämlich bereits die Fähigkeit zur Integration und wäre im RAMI4.0 in der Funktionsschicht anzusiedeln [80].

2. Im **Informationsbereich (Information Domain)** findet die Sammlung, Speicherung und Verarbeitung von Daten aus den anderen Funktionsbereichen statt, um Wissen über das gesamte System zu erhalten und dadurch Optimierung und Entscheidungsfindung des Systems zu beeinflussen [79]. Die entsprechende Informationsschicht im RAMI4.0 beschreibt außerdem semantisch Dienste und Daten [80], ansonsten erfüllt diese Schicht im RAMI4.0 die gleichen Basisfunktionen zur Datenverwendung.

3. Im **Betriebsbereich (Operations Domain)** sind die Funktionen zu Betrieb, Verwaltung, Überwachung und Optimierung des Steuerungsbereichs verortet [79].

Das Ziel dieses Bereichs ist die Sicherstellung der optimalen Leistung und Funktion des Systems [25]. Die entsprechenden Funktionen sind in RAMI4.0 in der Funktionsschicht abgebildet, in der alle Aufgaben zur Unterstützung der Geschäftprozesse erfolgen, zu welchen auch die Funktionen des Anwendungsbereichs der IIRA zählen [80].

4. Der **Anwendungsbereich (Application Domain)** dient zur Implementierung von Regeln, Modellen und anwendungsspezifischer Logik auf einer, im Gegensatz zum Betriebsbereich, höheren Ebene [79], um dort die Leistung und Effizienz des Systems zu verbessern [25].

5. Der **Geschäftsbereich (Business Domain)** bietet Funktionen zur Unterstützung von Geschäftsprozessen, um einen End-to-End Betrieb von IIoT-Systemen zu ermöglichen [79] und bestimmte Zielvorgaben zur Wertschöpfung zu erfüllen, während in der Geschäftsschicht im RAMI4.0 die Orchestrierung der Dienste der Funktionsschicht zur Integration in unterschiedliche Geschäftsprozesse und Geschäftsmodelle unter den rechtlichen Rahmenbedingungen erfolgt [80].

3.1.8 Zusammenfassung: Modelle zur Interoperabilität

In diesem Abschnitt wurden einige Modelle zu Interoperabilität von Systemen und Anwendungen vorgestellt. Die meisten dieser Modelle, insbesondere die früheren Modelle, wie das LCIM [57], [58], die Dimensionen der Interoperabilität nach ETSI [61] und IERC [52], das Semiotic Interoperability Framework [62], [63] und ISO/IEC 21823-1 [65] bauen auf der Semiotik bzw. der Wissenspyramide auf. Die Modelle IIRA [79] und RAMI4.0 [68]–[70] mit Bezug zu IIoT und I4.0 setzen auf vergleichbare Weise auf diesen eher abstrakten Konzepten auf, berücksichtigen aber zusätzliche technologische Aspekte, wie Querschittsfunktionen und Lebenszyklusbetrachtungen zur Anwendungsentwicklung.

Viele der Forschungsteams hinter diesen Modellen weisen außerdem auf die Komposition von IIoT bzw. I4.0 Anwendungen zu komplexen Systemen hin. Die aufwandsarme Komposition von solchen komplexen und zustandsbehafteten Anwendungen ist laut Neubacher et al. [77] bisher wenig erforscht. Bader et al. [73] beschreiben die Beziehungen zwischen Gegenständen als einen komplexen, zusammenhängenden Graphen von Automatisierungsgeräten mit einfachen Beziehungen zwischen Instanzen und Typen, Feedback-Schleifen und dynamischen, veränderlichen Beziehungen. Laut Bedenbender et al. [72] erfordert die Kooperation zwischen Geräten die Komposition von Geräten, die im aktuellen technischen Entwicklungsprozess statisch passiert, aber in I4.0 letztendlich dynamisch sein muss. Eine solche Komposition ist ein Komplex von Beziehungen.

Nach IIRA [79, S. 37] ist "das aggregierte Vehalten eines IIoT-Systems nicht die einfache Summe dessen, was von seinen einzelnen funktionalen Komponenten bereitgestellt wird"[6]. Einfache Beschreibungen von Verbindungen zwischen Geräten reichen ebenfalls nicht aus, die Eigenschaften dieser Beziehungen müssen ebenfalls formal beschrieben werden [72]. Weiters kann eine systemspezifische Logik bei solchen eng gekoppelten Systemen nicht in generischen Softwarekomponenten liegen [44], da alle Beziehungen anwendungsspezifisch und durch ihre Eigenschaften bestimmt sind und Geräte und Anwendungen eine gemeinsame Interaktionssprache, basierend auf diesen Eigenschaften, benötigen, um miteinander zu interagieren [72]. Die Bedeutung (Semantik) der Interaktionen zwischen Komponenten steht letztendlich in Abhängigkeit zum Transformationsverhalten [77], [78] und es bedarf folglich nicht nur Semantik, sondern auch des Zusammenspiels mit der Pragmatik, um Anwendungen und Geräte zu beschreiben.

3.2 Semantische Interoperabilität

Nach IEC21823-1 [65, S. 7] wird semantische Interoperabilität definiert als "Interoperabilität, bei der die Bedeutung des Datenmodells im Kontext eines Anwendungsgebiets von den teilnehmenden Systemen verstanden wird."[7] Nach Barnaghi et al. [28] bedeutet semantische Interoperabilität, dass "unterschiedliche Akteure auf Daten zugreifen und diese eindeutig interpretieren können."[8] Semantische Interoperabilität beschreibt also die Standardisierung auf Basis einer semantischen Beschreibung der kommunizierten Daten und dem damit einhergehenden eindeutigen Verständnis dieser Daten. Semantische Interoperabilität zielt daher darauf ab einheitliche Informationsmodelle zu schaffen, damit Geräte unterschiedlichster Quellen eine einheitliche Basis für das Verständnis haben. Serrano et al. [52, S. 6] definieren IoT als "das Netzwerk oder die Verbindungen zwischen über das Internet verbundenen Objekten, die Informationen über eine vereinbarte Methode und Datenschema austauschen können."[9] Daher bildet semantische Interoperabilität im Zusammenhang mit der Wissenpyramide (Abbildung 3.1) die Grundlage, damit aus ausgetauschten Daten Informationen werden. Aus Sicht der Forschung und Industrie kommen zunehmend Standardisierungsbemühungen zur Förderung der Interoperabilität auf, viele davon arbeiten an/fordern nach einer einheitlichen Semantik [27], [52], [81].

[6] "[..] the aggregate behavior of an IIoT system is not the simple sum of what is provided by its constituent functional components" [79, S. 37].

[7] "[...] interoperability so that the meaning of the data model within the context of a subject area is understood by the participating systems" [65, S. 7].

[8] "[...] different stakeholders can access and interpret the data unambiguously" [28, S. 3].

[9] "[...] the network or associations between those Internet connected objects (smart Devices) that are able to exchange information by using an agreed method and data schema" [52, S. 6].

Dafür gibt es bereits mehrere Ansätze, die im Bereich der Forschung üblicherweise genutzt werden, um einheitliche Informationsmodelle zu schaffen. Aus dem Bereich des Semantic Web [82] gibt es das Konzept von Ontologien, um Konzepte und deren Zusammenhänge abzubilden. Eine der bekanntesten Sprachen dafür ist die OWL Web Ontology Language (OWL) [83]. In der Industrie haben sich dagegen in den letzten Jahren um den Kommunikationsstandard OPC UA sogenannte Begleitspezifikationen (eng. "Companion Specifications") gebildet, um für unterschiedlichste Domänen einheitliche Informationsmodelle zu generieren [27].

3.2.1 Ontologien, Taxonomien & Vokabulare

Mit der Vision des *Semantic Web* durch Berners-Lee et al. [82] gab es bereits im Jahr 2001 die Idee, Webinhalten über Semantik eine Bedeutung zu verleihen. Dabei geht man auf das Problem von zentralisierten Systemen zur Wissensrepräsentation ein, welche die Notwendigkeit haben, dass nur exakt gleiche Konzepte zwischen Agenten ausgetauscht werden können, damit deren Bedeutung einheitlich ist. Laut den Autoren ist das Problem, dass so ein System bei steigender Größe nicht mehr handhabbar ist. Sie befürworten dagegen eine Menge verteilter Wissensrepräsentationen, um mehr Flexibilität zu erreichen, nehmen dabei aber auch Paradoxa in Kauf, die entstehen können, wenn formalisiertes Wissen aus unterschiedlichen Quellen stammt. Das Semantic Web soll demzufolge eine Erweiterung des Webs sein und dabei die bereits bestehende verteilte Infrastruktur des Internets um verteilte Semantik ergänzen, die von unterschiedlichsten Akteuren bereitgestellt und mit unterschiedlichsten Ressourcen verknüpft werden kann. Diese verteilte Semantik wird im Semantic Web durch verteilte Ontologien abgebildet. Nach Gruber [84, S. 1] ist eine Ontologie "eine explizite Spezifikation eines Konzepts."[10] Diese beschreibt mit einer Menge repräsentativer Begriffe die in diesem Konzept vorhandenen Objekte, deren Zusammenhänge und Axiome, die die Interpretation einschränken.

Ontologien lassen sich klar von Vokabularen und Taxonomien unterscheiden. Nach McComb [85] ist ein Vokabular eine Menge an Wörtern, also Zeichen, denen in einer Sprache eine einheitliche Bedeutung verliehen wird. Eine Taxonomie ist darauf aufbauend ein klassischerweise hierarchisch organisiertes Vokabular. Eine der bekanntesten Taxonomien ist die Systematik der Lebewesen aus der Biologie. Der Unterschied zwischen Taxonomien und Ontologien ist, dass eine Ontologie zwar auch hierarchische Strukturen aufweisen kann, zusätzlich aber weitere Beziehungen zwischen den einzelnen Konzepten aufweist. Damit sind Ontologien meist Graphen oder Netzwerke, die mit

[10] "An ontology is an explicit specification of a conceptualization" [84, S. 1].

Beziehungen, Bedingungen und Regeln verknüpft sind, mit denen Schlussfolgerungen (*Reasoning*) aus der Ontologie gezogen werden können [85].

Aus der Sicht des Semantic Web ist eine Ontologie ein Dokument, das die Beziehung zwischen Begriffen definiert [82]. Dazu wird der Resource Description Framework (RDF) [86] Standard verwendet, um mit einer Kombination aus Subjekt, Prädikat und Objekt die Begriffe und die Beziehungen zwischen diesen Begriffen zu beschreiben. Alle verwendeten Begriffe sind dabei eindeutig über Unique Resource Identifier (URI) oder Unique Resource Locator (URL) identifizierbar. Als Basis für die RDF Sytax wird nach Berners-Lee eXtensible Markup Language (XML) [87] verwendet, es werden aber auch andere Beschreibungssprachen unterstützt. Als Metaontologien haben sich das RDF Schema (RDFS) [88] und OWL [83] etabliert, um Ontologien zu definieren und einfache Zusammenhänge zu inferieren.

Nachdem das Semantic Web von einem verteilten Set an Ontologien ausgeht, werden Mechanismen benötigt, um Daten aus unterschiedlichen Quellen zu kombinieren und diesen eine Bedeutung zuzuordnen. Dabei wird einerseits davon ausgegangen, dass Beziehungen zwischen unterschiedlichen Ontologien bestehen und eine Anwendung über diese Beziehungen den Zusammenhang mit einer vorher unbekannten Ontologie inferieren kann. Andererseits können neue Zusammenhänge zwischen unterschiedlichen Ontologien über *Ontology Alignment*, also einen Abgleich von Begriffen und Strukturen, erschlossen werden. Da eine deterministische Beziehung zwischen Ontologien aber einen sichereren Zusammenhang liefert, ist es bei der Entwicklung einer Ontologie wichtig, viele Konzepte aus anderen Ontologien wiederzuverwenden. Laut McComb [85] ist eine obere Ontologie (*upper ontology*) als Basis hilfreich, damit Anwendungen domänenübergreifend Beziehungen zu anderen Ontologien inferieren können. Auch nach DIN/DKE [23] werden solche domänenspezifischen top-level Ontologien vorgezogen, um eine gemeinsame Grundlage zu bilden.

3.2.2 OPC Unified Architecture Companion Specifications

Mit dem Kommunikationsstandard OPC UA [89] hat sich die Bedeutung der Semantik für die Kommunikation als integraler Bestandteil herauskristallisiert. Im Gegensatz zu den Vorgängerversionen innerhalb der Open Platform Communications (OPC) (früher: OLE for Process Control (OPC)) Standards, zu deren Bekanntesten OPC Data Access (OPC DA) zählt, konzentriert sich der UA Standard auf Plattformunabhängigkeit und Interoperabilität. Laut Mahnke et al. [26] ist die Interoperabilität zwischen Systemen von unterschiedlichen Herstellerfirmen die wichtigste Anforderung von OPC UA, neben Robustheit, Skalierbarkeit, Sicherheit und Zugangskontrolle.

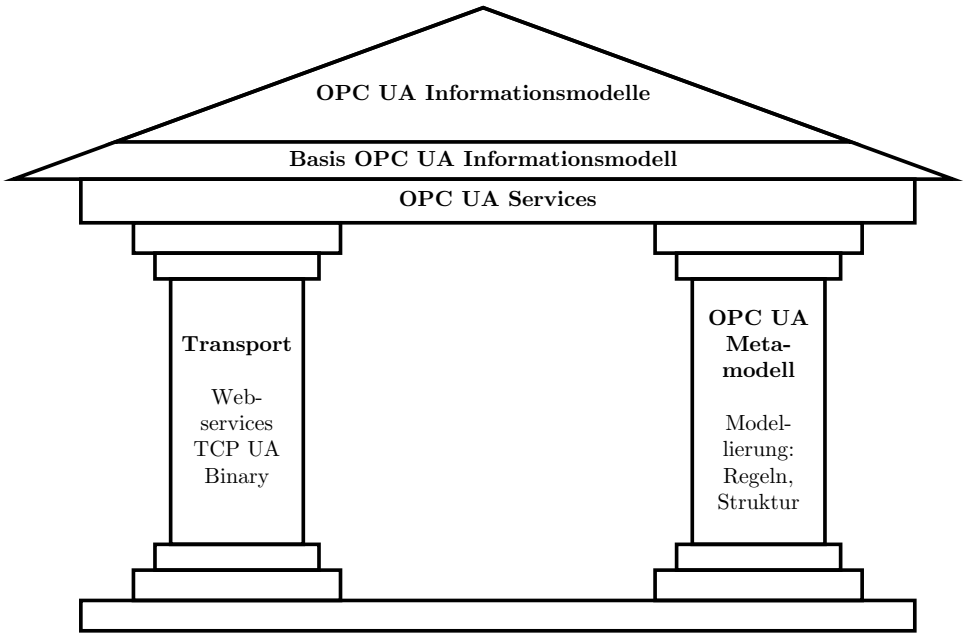


Abbildung 3.7: OPC UA Übersicht [26].

OPC UA baut neben der Möglichkeit unterschiedliche Transportprotokolle, wie Web Services und ein binäres Protokoll (TCP UA Binary), zu nutzen (siehe Abbildung 3.7) auch darauf, dass von einem standardisierten Metamodell und Basis Informationsmodell aus unterschiedliche Informationsmodelle aus verschiedenen Domänen objektorientiert modelliert, standardisiert und bei der Kommunikation referenziert werden können. Diese Informationsmodelle entsprechen im weitesten Sinne dem Konzept von Ontologien, da sie eine graph-basierte Modellierung mit Knoten und Referenzen zwischen diesen Knoten erlauben. OPC UA weist unter anderem eine klassische Server-Client Architektur auf, über die der Server dem Client nicht nur Daten, sondern auch Informationen, die mit der Semantik der Informationsmodelle verknüpft sind, bereitsstellen kann.

Neben dem dezentralen Ansatz des Semantik Web verfolgt OPC UA einen eher zentralen Ansatz, bei dem Informationsmodelle in gemeinsamen Arbeitsgruppen (*Joint Working Groups*) von Domänenexperten wie Wissenschaftlern und Branchenführern gemeinsam entwickelt werden und als sogenannte Begleitspezifikation (*Companion Specification*) veröffentlicht werden können, welche heruntergeladen und zur Entwicklung semantisch valider Informationsmodelle herangezogen werden können. Daneben gibt es aber auch die Möglichkeit externe Arbeitsgruppen zu bilden. Zur Zeit gibt es 54 offizielle Arbeitsgruppen, die an unterschiedlichsten Begleitspezifikationen arbeiten [27]. Graube et al. [90] kritisieren daran, dass es noch keine freien Verteiler für Informati-

onsmodelle gibt und der Aufwand eine OPC UA Arbeitsgruppe zu erstellen so hoch ist, dass die agile, kollaborative Entwicklung von Informationsmodellen dadurch behindert wird. Entgegen der aktuell zentralisierten Art der Informationsmodellentwicklung der OPC Foundation fordern die Autoren einen Community-getriebenen schrittweisen Standardisierungsprozess. Ein solcher Prozess zur Entwicklung von Informationsmodellen auf Basis von verteilten Ontologien wie im Semantic Web ist durchaus denkbar und einige Foscher, wie Perzylo et al. [91] haben sich bereits mit der Übersetzung zwischen OPC UA und OWL beschäftigt. Dagegen kritisieren Profanter et al. [92], dass durch proprietäre Erweiterungen von Informationsmodellen die Austauschbarkeit von Geräten verschiedener Herstellerfirmen schwieriger wird.

3.3 Vom Semantic Web zum Pragmatic und Dynamic Web

In den Jahren nach der Entstehung und Entwicklung des Semantic Web (2001) kamen immer mehr Diskussionen über dessen Herausforderungen auf [93] und ob das Semantic Web eine ausreichende Lösung bietet, um Wissen in unabhängigen Ontologien sinnvoll miteinander zu verbinden.

Bravo und Velazquez [94] kritisieren am Semantic Web, dass es hauptsächlich für die Integration verschiedenartiger Datenquellen, wie verteilten Datenbanken und Ontologien, verwendet wird. Sie führen weiter aus, dass das Internet nicht nur aus Daten- und Informationsquellen besteht, für die dieser semantische Ansatz ausreichen würde, sondern auch aus Programmen, Agenten, Webanwendungen und Diensten, die als interaktive Softwareagenten miteinander über Datenaustausch, Protokolle oder Regeln interagieren müssen. Tolk [95] kritisiert, dass das Semantic Web zwar gemeine Sichtweisen und Situationen beschreibt, aber zur Orchestrierung von dynamischen und voneinander abhängigen Anwendungen höhere Ebenen der Interoperabilität notwendig sind, die nicht nur die Semantik der übertragenen Daten abbilden, sondern auch Pragmatik, Dynamik und Konzepte. Viele Forschungsteams, die sich mit dem Thema der pragmatischen Interoperabilität beschäftigen, argumentieren, dass Syntax und Semantik nicht ausreichen, um eine bedeutungsvolle Interoperabilität zu erreichen [66], [93], [94], [96]–[99]. Diese Sichtweise lässt sich wahrscheinlich mit dem durch das Semantic Web hervorgerufenen Hype zur semantischen Beschreibung von Ressourcen begründen und der teilweise unrealistischen Erwartungshaltung, dass das Semantic Web eine Lösung vieler Interoperabilitätsprobleme liefern sollte.

Pragmatic Web

De Moor et al. [97] kritisieren, dass viele Enthusiasten des Semantic Web es scheinbar für selbstverständlich erachten, dass sich Semantik von selbst um die Entwicklung von Wissen kümmert. Dabei fehlt jedoch mit der Pragmatik auch der Zweck der Informationen. Die Autoren schlagen daher die Entwicklung des *Pragmatic Web* vor, in dem pragmatische Prozesse definiert und automatisiert sind. Nach de Moor [98] besteht es desweiteren neben dem Semantik Web als eine Menge pragmatischer Kontexte von den semantischen Ressourcen des Semantic Web, wobei ein pragmatischer Kontext von den Definitionen und Interessen einer Domäne (Community) geprägt ist. Dabei sind Semantic und Pragmatic Web konzeptuell voneinander getrennt, aber voneinander abhängig. Der Autor schlägt weiter vor, dass sogenannte *pragmatische Patterns*, die Komponenten zur Problemlösung in einem bestimmten Kontext enthalten, wie sie auch der Mensch nutzt, um komplexe Sachverhalte zu verstehen, verwendet werden. Diese sollen eine andere Sichtweise auf eine Ontologie ermöglichen. Das Pragmatic Web ist nach Schoop et al. [100] eine Weiterentwicklung des Semantic Web in dem die menschliche Kooperation bei der Entwicklung von Ontologien durch geeignete Technologien unterstützt wird, um Ontologien zu verhandeln oder eine gemeinsame Ontologieentwicklung in Domänen zu unterstützen. Die Autoren begründen diese Notwendigkeit damit, dass Ontologien nie statisch sind, sondern sich immer innerhalb einer Domäne weiterentwickeln.

Eine ähnliche Entwicklung des Pragmatic Web schlägt auch Singh [99], [101] vor, der in der Pragmatik den Erfolg für das Semantic Web sieht und die **Designprinzipien für das Pragmatic Web** liefert:

1. **Nutzer vor Anbieter (user before provider):** Die Modellierung der kontextabhängigen Bedürfnisse des Nutzers ist gleich oder sogar wichtiger als die Modellierung der Fähigkeiten des Anbieters.

2. **Prozess vor Daten (process before data):** Der Kontext in dem Daten erzeugt oder verwendet werden ist wichtig, aber schwer zu erfassen. Jedoch hat die Person, die die Prozesse modelliert ein besseres Verständnis von der Bedeutung der Daten als die Person, die die Daten ohne den Zusammenhang zu den Prozessen sieht.

3. **Interaktion vor Repräsentation (interaction before representation):** Genauso wie funktionale Schnittstellen für Datenstrukturen deren Implementierungsdetails verbergen, verbergen auch Service-Interaktionsmodelle die "überschüssige" Semantik.

Dynamic Web

Tolk [95] beschreibt eine Entwicklung zum *Dynamic Web* (web of composable services) als eine Weiterentwicklung des Semantic Web (web of data). Der Autor geht dabei zwar spezifisch auf die Domäne der verteilten Simulation im Bereich Modellierung und Simulation (MS) ein und beschreibt, dass hierfür das Semantic Web nicht ausreicht, da in dieser Domäne die Orchestrierung und Angleichung von hoch-agilen und dynamischen, ineinandergreifenden Anwendungen notwendig ist. Diese Argumente sind aber nicht nur im Bereich MS, sondern generell für die Interaktion von verteilten Anwendungen und Webservices gültig. Ein Unterschied zum Ansatz des Pragmatic Web ergibt sich durch die Anwendung des Schichtenmodells der konzeptuellen Interoperabilität (eng. LCIM [57], [58]), das über die pragmatische Interoperabilität hinaus noch weitere Schichten einschließt, wie die dynamische und konzeptuelle Interoperabilität (Abbildung 3.3). Die höheren Ebenen der Interoperabilität benötigen zu den ausgetauschten Daten neben deren Semantik ebenfalls die Möglichkeit zur Komposition (über die Pragmatik), den Einfluss des Datenaustauschs auf Sender und Empfänger (Dynamik) und die Bedingungen für die Komposition (Konzepte) [95]. Die Grundlage für die Vision dieses Dynamic Web bilden verteilte Webservices, die sich über das Internet finden und gegenseitig verwenden können. Über die Komposition können schließlich verteilte Webservices zusammen interagieren, um einen gewissen Zweck zu erfüllen. Laut Tolk [95] ist für die Komposition eine konzeptuelle Schicht notwendig, die Annahmen und Bedingungen enthält. Diese Schicht stellt durch ein vollständig spezifiziertes, konzeptuelles Modell den Zusammenhang zu den darunterliegenden Schichten und Daten, unabhängig vom ausgeführten Code, her [57], [59]. Folglich ist es eine abstrakte Beschreibung von Sachverhalten und notwendigen Bedingungen, über die eine Zieldefinition für die verteilten Services erfolgt.

3.4 Pragmatische Interoperabilität

Asuncion und Sinderen [66] vergleichen in einer Studie die unterschiedlichen Definitionen von pragmatischer Interoperabilität und finden Unterschiede zwischen zwei Gruppen, bei denen pragmatische Interoperabilität entweder ein gemeinsames Verständnis und eine gemeinsame Erwartungshaltung beim Nutzen von Daten in einem bestimmten Kontext bedingt, oder ein gemeinsames Verständnis von der Verwendung von Diensten liefert. Die Autoren selbst definieren pragmatische Interoperabilität als die Kompatibilität zwischen dem vorgesehenen und dem tatsächlichen Effekt des Nachrichtenaustauschs [66], [67]. In einer Literaturstudie im Jahr 2016 zu pragmatischer Interoperabilität haben Neiva et al. [93] herausgefunden, dass die meisten Lösungen

in den Bereichen *e-Business*, *World Wide Web* und *Health* angewandt wurden und die meisten Artikel die Aspekte *Servicekomposition*, *Service Discovery* und *Ontologien* behandelten. Obwohl laut den Autoren keiner der untersuchten Artikel pragmatische Interoperabilität durch eine Evaluierung von Experimenten oder Fallstudien mit klar definierten Hypothesen, Methoden und statistischen Analysen untersucht hat, haben viele Studien die Notwendigkeit von pragmatischer Interoperabilität beschrieben. Sie schlussforgern daraus die Notwendigkeit weiterer Studien auf diesem Forschungsgebiet. Hervorzuheben ist auch, dass Neiva et al. [93] keine Arbeiten im Bereich Produktion und Fertigung gefunden haben.

3.4.1 Konzepte pragmatischer Modellierung

Es gibt viele Formalismen, um pragmatische Zusammenhänge abzubilden. Bravo et al. [94] zählen einige davon auf: *Petri Nets*, *Colored Petri Nets* [102], *Pi-calculus*, *Agent Unified Modeling Language (AUML)*, *Business Process Execution Language (BPEL)*, *OWL Web Ontology Language for Services (OWL-S)* und endliche Zustandsautomaten, auch bekannt als *Finite State Machine (FSM)*. Jedoch haben laut den Autoren alle dieser Formalismen unterschiedliche Zwecke: Petri Nets, Colored Petri Nets, Pi-Calculus und AUML dienen der formalen Beschreibung von Interaktionen, Petri Nets und Colored Petri Nets dienen zusätzlich dazu auch der Verifikation und Simulation, BPEL und OWL-S dienen der Komposition von Diensten. Bravo et al. [94] verwenden in ihrem Artikel FSM zur Beschreibung der Interaktionen zwischen Agenten, da diese eine einfache Methode zur Implementierung von Übergangsfunktionen bieten. Die IEC21823 [65] nennt als Beispiele für verhaltensbasierte Interoperabilität (behavioral interoperability) Unified Modelling Language (UML) Modelle, womit wahrscheinlich auf die Menge der Verhaltensdiagramme in UML (u.a. Aktivitäts-, Anwendungsfall- und Zustandsdiagramm) angespielt wird. Die bekannteren Konzepte zur pragmatischen Modellierung sind Automaten. Diese stellen in der Informatik Modelle dar, die das Verhalten eines Systems beschreiben. Zu den bekanntesten Automaten zählt das von Alan Turing [103] entwickelte Modell der Turingmaschine. Endliche Automaten (FSM) dagegen sind ein Spezialfall von Automaten, die das Verhalten eines komplexes Systems mit einer endlichen Anzahl an Zuständen und Zustandsübergängen beschreiben. Da im weiteren Verlauf dieser Arbeit FSM mit UML-ähnlichen Zustandsdiagrammen in State Chart XML (SCXML) abgebildet werden, weil diese eine besonders einfache und grafische Notation erlauben und in ähnlicher Form auch bereits in OPC UA abgebildet sind, werden diese im Folgenden beschrieben.

3.4.2 Endliche Zustandsautomaten

Zu den ersten endlichen Zustandsautomaten (FSM) zählen die Automaten von Mealy [104] im Jahr 1955 und Moore [105] im Jahr 1956. Diese endlichen Automaten, oft dargestellt als Set von $(Q, q_0, X, Y, \delta, \lambda)$, beschreiben das Verhalten von Systemen mit:

- einer endlichen Menge Q an Zuständen: $q_i \in Q$
- einem Startzustand: $q_0 \in Q$
- einer endlichen Menge X an Eingabesymbolen: $x_i \in X$
- einer Menge Y an Ausgabesymbolen: $y_i \in Y$
- der Zustandsübergangsfunktion $\delta : Q \times X \rightarrow Q$ bzw. in Form von $p = \delta(q, x)$ mit $p, q \in Q$ [106], [107]
- der Ausgabefunktion $\lambda : Q \times X \rightarrow Y$ (Mealy) oder $\lambda : Q \rightarrow Y$ (Moore)

Die beiden Varianten unterscheiden sich nur hinsichtlich der Ausgabefunktion, da diese bei Mealy-Automaten abhängig vom Zustand und der Eingabe ist und bei Moore-Automaten nur abhängig vom Zustand. In Abbildung 3.8 wird der Unterschied deutlich, da im Mealy-Automaten die Ausgabesymbole y_i in Abhängigkeit von den Zustandsübergängen und im Moore-Automaten in Abhängigkeit von den Zuständen dargestellt werden.

Zustandsübergangsfunktionen:
$\delta_0 : q_1 \times x_0 \rightarrow q_0$
$\delta_1 : q_0 \times x_1 \rightarrow q_1$
$\delta_2 : q_1 \times x_1 \rightarrow q_1$
Ausgabefunktionen (Moore):
$\lambda_0 : q_1 \rightarrow y_0$
$\lambda_1 : q_0 \rightarrow y_1$
Ausgabefunktionen (Mealy):
$\lambda_0 : q_1 \times x_0 \rightarrow y_0$
$\lambda_1 : q_0 \times x_1 \rightarrow y_1$
$\lambda_2 : q_1 \times x_1 \rightarrow y_2$

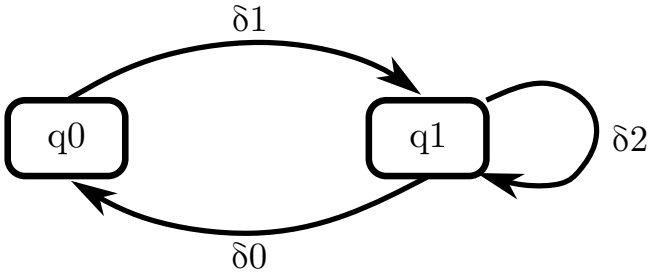
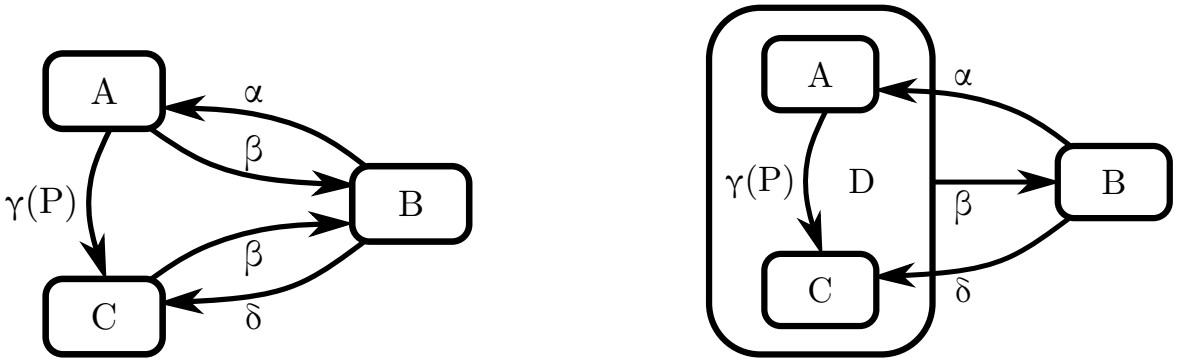


Abbildung 3.8: Moore- vs. Mealy-Automat.

Da die Zustände im Mealy-Automaten damit flexibler darstellbar sind, erlaubt dieser eine einfachere Darstellung komplexer Systeme, als der Moore-Automat. Da diese



(a) Flacher Zustandsautomat

(b) Hierarchischer Zustandsautomat

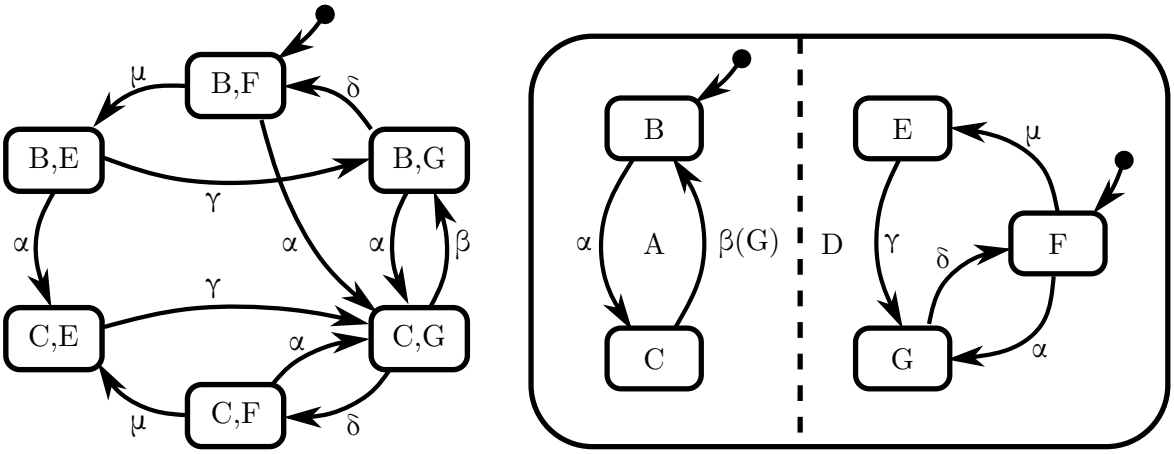
Abbildung 3.9: (a) Flacher Zustandsautomat im Vergleich zu seinem (b) hierarchischen Pendant [108].

bis dahin flachen Zustandsdiagramme (State Diagram) aber laut Harel [108] nicht geeignet sind, um komplexe Systeme handhabbar zu beschreiben, da deren Größe exponentiell wächst, schlägt er unter dem Begriff *State Chart* eine neue Notation und später auch deren Semantik [109] vor. Harel [108] verwendet drei Konzepte, um die Zustände einzuschränken und die Komplexität zu reduzieren: Orthogonalität, Hierarchie und Kommunikation.

> "However, it is also generally agreed that a complex system cannot be beneficially described in this naive fashion, because of the unmanageable, exponentially growing multitude of states, all of which have to be arranged in a 'flat' unstratified fashion, resulting in an unstructured, unrealistic, and chaotic state diagram. To be useful, a state/event approach must be modular, hierarchical and well-structured." - David Harel [108, S. 233]

Hierarchie hilft ähnliche Zustände logisch unter einem übergeordneten Zustand zu vereinen und dadurch gemeinsame Zustandsübergänge (β in Abbildung 3.9) zu reduzieren. Die gemeinsamen Eigenschaften von Zuständen, die in Form von Zustandsübergängen vorliegen, werden dabei durch einen solchen abstrakten Zustand (D in Abbildung 3.9) gekapselt.

Orthogonalität erlaubt es nebenläufige Systemzustände, wie in Abbildung 3.10 (a) (Zustände $B, C \in A$ und $E, F, G \in D$) innerhalb von orthogonalen Subzuständen (A, D) aufzuteilen und dabei die Anzahl an Zuständen und Zustandsübergängen stark zu reduzieren (Abbildung 3.10 (b)). Bei einer geringen Anzahl von orthogonalen Systemzuständen, wie in Abbildung 3.10 (a), ist der Vorteil durch die Modellierung



(a) Flacher Zustandsautomat mit orthogonalen Zuständen

(b) Orthogonaler Zustandsautomat

Abbildung 3.10: (a) Flacher Zustandsautomat im Vergleich zu seinem (b) orthogonalen Pendant [108].

nebenläufiger Zustände begrenzt. Mit zunehmender Menge an orthogonalen Zuständen wächst die Anzahl der Optionen jedoch exponentiell. Das daraus resultierende Problem wird als Zustandsexplosion (eng. "state explosion problem") beschrieben. Clarke et al. [110] beschreiben beispielsweise ein System, das aus n Prozessen besteht, die jeweils m Zustände haben. Die daraus resultierende asynchrone Kombination dieser Prozesse hat dann bereits m^n Zustände. Folglich gilt für die Gesamtanzahl von Zuständen aus der Kombination einer Menge von orthogonalen Zuständen P mit einer jeweils unterschiedlich großen Menge an atomaren Subzuständen von $P_n \subset P$, dass $\prod_n^{|P|} |P_n|$.

Kommunikation wird von Harel [108] letztendlich durch die Notwendigkeit einer Kooperation zwischen orthogonalen, nebenläufigen Zuständen begründet. Diese besitzen die Notwendigkeit auf gemeinsame Variablen zuzugreifen oder Zustandsänderungen zu kommunizieren und damit weg von den einzelereignisbasierten Zustandsübergängen zu einer Sammelmeldung von Ereignissen zu kommen. Abbildung 3.10 zeigt die Abhängigkeiten von Ereignissen in orthogonalen Zuständen, aber auch die Schwierigkeit ohne klar definierte Semantik einer klare Reihenfolge von Zustandsübergängen zu erkennen.

Zustandsdiagramme nach Harel sind damit eine Weiterentwicklung endlicher Automaten, mit dem komplexe Systeme modelliert werden können [111]. Die im UML [112] Standard durch die Object Management Group (OMG) standardisierte Form endlicher Zustandsautomaten basiert auf einer objektorientierten Variante dieser Zustandsdiagramme, die nur durch kleine Unterschiede in der zugrundeliegende Semantik

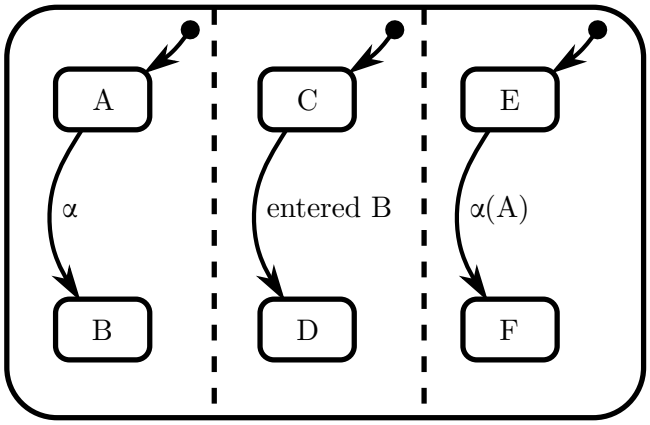
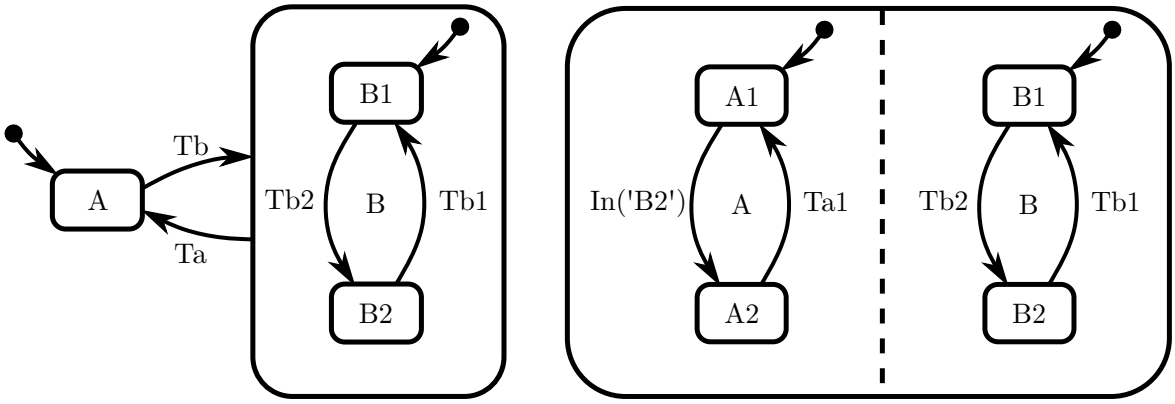


Abbildung 3.11: Kommunikation zwischen orthogonalen Zuständen [108].

differenziert werden. In UML sind Zustandsautomaten darauf reduziert das Verhalten eines Systems zu beschreiben, während die Zustandsdiagramme nach Harel auch die Struktur beschreiben [111]. Neben der Modellierung des ereignisbasierten Verhaltens eines Teilsystems, können in UML außerdem zulässige Interaktionsabläufe in Form von Protokollen modelliert werden [112]. Eine Erweiterung von UML Zustandsautomaten gibt es in der Sprache Systems Modeling Language (SysML) [113], die ebenfalls von der OMG standardisiert wurde. Nach Berardinelli et al. [114] ist SysML von UML für die Entwicklung großer, komplexer und multidisziplinärer Systeme abgeleitet und verwendet einige Elemente aus UML wieder, besitzt aber auch neue Komponenten. Zustandsautomaten in SysML entsprechen denen in UML, jedoch ohne die Erweiterung um Protokoll-Zustandsautomaten.



(a) Hierarchie (b) Orthogonalität

Abbildung 3.12: (a) Hierarchie und (b) Orthogonalität von Zuständen in State Chart XML (SCXML).

```
<scxml name="hierarchical" initial="A">
  <datamodel>
    <data id="count" expr="0"/>
  </datamodel>
  <state id="A">
    <transition event="Tb" target="B"/>
  </state>
  <state id="B">
    <initial>
      <transition target="B1"/>
    </initial>
    <state id="B1">
      <transition event="Tb2" target="B2"/>
    </state>
    <state id="B2">
      <transition event="Tb1" target="B1"/>
    </state>
    <transition event="Ta" target="A"/>
      <assign location="count" expr="count + 1"/>
    </transition>
  </state>
</scxml>
```

Codeabschnitt 3.1: Hierarchie von Zuständen in State Chart XML (SCXML).

Eine weitere Form von Zustandsautomaten ist mit der Notation SCXML [115] abgebildet, die sich selbst auf der Basis der Notation von Zustandsdiagrammen nach Harel und der Syntax für Zustandsautomaten und Elemente nach Call Control XML (CCXML) [116] versteht. Das Ziel des Standards ist die Semantik von Harel mit einer XML Syntax nach CCXML zu kombinieren [115]. SCXML besitzt demnach Elemente, um Hierarchie, Parallelität und Kommunikation mittels XML zu beschreiben, und eine Semantik, die definiert, wie diese auszuführen ist. **Hierarchie**, wie in Abbildung 3.12 (a), wird hier über die Ebenen von *state* Elementen in der XML Baumstruktur definiert (Codeabschnitt 3.1). Diese Zustände können weitere Elemente besitzen, wie tiefere Zustände oder Zustandsübergänge (*transition*). Zustand B besitzt dabei die beiden Zustände $B1$ und $B2$: $B1, B2 \in B$. Die Transition Tb bewirkt dabei einen Zustandsübergang von Zustand A auf Zustand $B1$: $A \times Tb \rightarrow B1$. Ta kann entweder von Zustand $B1$ oder $B2$ aus einen Zustandsübergang zu A bewirken: $(B1 \vee B2) \times Tb \rightarrow A$.

Orthogonalität (in SCXML als **Parallelität** bezeichnet), wie in Abbildung 3.12 (b), wird über ein *parallel* Element in XML definiert (Codeabschnitt 3.2). Unter dem Zustand P sind hier die beiden orthogonalen Zustände A und B definiert, die bei einem Zustandsübergang in Zustand P beide aktiv werden.

Kommunikation ist in SCXML einerseits durch das *datamodel* Element und die darin enthaltenen *data* Variablen (Codeabschnitt 3.1, Zeilen 3-5) für den von Harel [108] geforderten gemeinsamen Zugriff auf Variablen dargestellt. Zusätzlich existieren Elemente und Attribute, die entweder den Zuständen (*onentry, onexit*) oder Zustandsübergängen (*assign, cond* in Codeabschnitt 3.1, Zeile 20) zugeordnet sind, mit denen auf bestimmte Ereignisse, auch zwischen orthogonalen Zuständen (Codeabschnitt 3.2, Zeile 11), reagiert werden kann.

```
1  <scxml name="parallel" initial="P">
2    <datamodel>
3    </datamodel>
4    <parallel id="P">
5      <state id="A">
6        <initial>
7          <transition target="A1"/>
8        </initial>
9        <state id="A1">
10         <transition cond="In('B2')" target="A2"/>
11       </state>
12       <state id="A2">
13         <transition event="Ta1" target="A1"/>
14       </state>
15     </state>
16     <state id="B">
17       <initial>
18         <transition target="B1"/>
19       </initial>
20       <state id="B1">
21         <transition event="Tb2" target="B2"/>
22       </state>
23       <state id="B2">
24         <transition event="Tb1" target="B1"/>
25       </state>
26     </state>
27   </parallel>
28 </scxml>
```

Codeabschnitt 3.2: Parallele Zustände in State Chart XML (SCXML).

3.4.3 Skills

Die Plattform I4.0 [53] beschreibt die Fähigkeiten (Skills) einer I4.0 Komponente als deren *"Potenzial [...], eine Wirkung in einer Domäne zu erzielen."* Diese lassen sich mit

dieser Definition ebenfalls in die Methoden pragmatischer Interoperabilität einordnen, so lange der Fokus darauf liegt, dass ein bestimmter Effekt erzielt wird. Besonders in der Forschung hat sich dazu in den letzten Jahren das Konzept *Skill-based Engineering* verbreitet. Zimmerman et al. [117] beschreiben, dass sich Skill-based Engineering mit dem Entwurf von Automatisierungssystemen befasst und diese auf der Grundlage einer adäquaten Orchestrierung der notwendigen Fähigkeiten basieren. Nach Malakuti et al. [118] ist das Hauptziel von Skill-based Engineering, Schritte in Produktionsprozessen abhängig von den benötigten Fähigkeiten anstatt der tatsächlich verwendeten Ressourcen beschreiben zu können. Damit erlaubt das Konzept, Fertigungsprozesse mittels abstrakten Skills und deren Anforderungen zu formulieren und tatsächliche Ressourcen über deren angebotene Skills auszuwählen, was die Fertigungsprozesse zur Laufzeit flexibel austauschbar und leicht skalierbar macht. Skill-based Engineering bietet damit zum pragmatischen Ansatz von Skills eine passende konzeptuelle Ebene.

Basierend auf dem Produkt-Prozess-Ressource (PPR) Modell [119] und dessen Erweiterung von Björkelund et al. [120] teilen Malakuti et al. [118] Skills in ein vierdimensionales Modell ein, das diese nach ihrem Kompositionsgrad und ihrer Beziehung zu PPR klassifiziert. Folglich können Skills unabhängig bis stark abhängig von einem bestimmten Einsatzbereich und einfach bis sehr komplex sein. Die Klassifizierung bestimmt, wie leicht ein Skill in einem anderen Kontext wiederverwendet werden kann. Zum Beispiel sind atomare Skills einer Werkzeugmaschine, wie das Öffnen einer Beladetür, oft stark abhängig von Prozess und Ressource, aber weniger vom Produkt. Daneben hat ein Skill zum Beladen eines Werkstückes in eine Werkzeugmaschine schon einen höheren Kompositionsgrad, da dies mehrere andere Skills einschließt und bei einer direkten Werkstückhandhabung auch produktabhängig ist.

Um Skill-based Engineering umzusetzen zu können, fordern viele Forschungsteams eine Standardisierung von Skills [117], [118], [121]. Malakuti et al. [118] argumentieren, dass bestehende Klassifikationsschemat und reine Vokabulare nicht ausreichen, um die Semantik von Skills zu beschreiben und fordern eine Standardisierung durch Organisationen. Auch Perzylo et al. [121] fordern speziell die Standardisierung der Semantik von Skills. Trotz einer Standardisierung von Skills können letztendlich auch proprietäre Funktionen mit einem abstrakten Skillmodell verwendet [92] und so die Anforderungen der Industrie an eine flexiblere Standardisierung erfüllt werden.

3.4.4 Servicekomposition

Die Komposition von Services wird in vielen Forschungsartikeln [93], [95], [99] auf diesem Gebiet als einer der Kernpunkte genannt, der durch pragmatische Interoperabilität

ermöglicht wird. Daher kommt die pragmatische Interoperabilität als möglicher Ansatz für die Interaktion von verteilten Diensten und Programmen in Frage. Singh [99] beschreibt die automatische Komposition von Webservices daher auch als die Kernherausforderung im Pragmatic Web, da dadurch aus vorhandenen Services neue Services entstehen können: Da Webservices autonome, heterogene, miteinander interagierende Anwendungen sind, werden zur Komposition Abstraktionen dieser Eigenschaften benötigt, die auch die Bedingungen für die Interaktion zwischen den Services abbilden. Auch nach Tolk [95] ist die Fähigkeit zur Komposition von Services auf abstrakter Ebene für eine substanzielle Interoperabilität notwendig. Laut Singh [99] werden Modelle mit semantischen und pragmatischen Bedingungen benötigt, um zu beschreiben, wie Services in verschiedenen Kompositionen miteinander interagieren können und außerdem die Validität dieser Kompositionen überprüfen zu können. Weiter sollen auch mögliche Fehlerzustände in Kompositionen durch eine ausdrucksstarke Beschreibung der semantischen und pragmatischen Bedingungen hervorgehoben werden.

3.5 Die Entwicklung der Interoperabilität

Bereits in den unterschiedlichen Modellen zu Interoperabilität spiegelt sich der Trend zur Kombination der semantischen mit der pragmatischen Interoperabilität. Auch die Entwicklungen im Semantik Web zeigen, dass semantische Ansätze schon in der Vergangenheit und nicht nur heute intensiv genutzt wurden, um verteile Services über Beschreibungen mit einer einheitlichen Semantik leichter miteinander zu verknüpfen. Dabei hat es aber schon früh Kritik gegeben, ob diese semantischen Beschreibungen alleine ausreichen, um die vollständige Interoperabilität zwischen Services zu ermöglichen. Im Jahr 2009 haben Wang et al. [60] herausgefunden, dass sich viele wissenschaftliche Artikel entweder mit den unteren Ebenen der Interoperabilität (syntaktische und semantische Interoperabilität) oder mit den abstrakteren Ebenen (konzeptuelle Interoperabilität) beschäftigen, aber die pragmatischen und dynamischen Ebenen vernachlässigt werden. Wenngleich auch heutzutage in den Modellen zur Interoperabilität und offiziellen Kommissionen und Gremien zumindest bekannt ist, dass diese Ebenen existieren [52], [65] und Interoperabilität auch mit der Wissenpyramide assoziiert wird [23], die die Pragmatik als wesentliche Komponente für die Entstehung oder Beschreibung von Wissen aus zweckgebundenen Informationen [54] festlegt, so sind aktuelle praktische Entwicklungen zum Fortschritt der Interoperabilität, wie aktuelle OPC UA Begleitspezifikationen [27], weiterhin hauptsächlich auf die von Wang et al. [60] beschriebenen Ebenen fokussiert.

Kapitel 4

Implikationen im Rahmen der Arbeit

Aus dem vorherigen Kapitel 2 lässt sich mit dem Trend zu agil-rekonfigurierbaren Fertigungssystemen die Notwendigkeit zur flexiblen Automatiserbarkeit von Fertigungssystemen und deren Komponenten, u. a. durch die Entwicklung von CPPS, erkennen. Dagegen zeichnen sich in Kapitel 3 die Schwierigkeiten, mit denen solche Systeme derzeit bei der Integration zu kämpfen haben, ab. Die Landschaft an Komponenten (wie Maschinen, Geräten und softwaretechnischen Anwendungen) solcher Systeme ist geprägt von Heterogenität [52] und Komplexität der unterschiedlichen Komponenten und deren systemspezifischen Abhängigkeiten untereinander. Fertigungssysteme können durch die Entwicklung zu CPPS ebenfalls wie verteilte Systeme betrachtet werden. Dabei können Funktionen von Maschinen als verteilte Services angesehen werden und benötigen die Fähigkeit zur Kooperation in diesem Fertigungssystem.

4.1 Forschungsfragen

Aufgrund der in Kapitel 3, insbesondere Abschnitt 3.3, identifizierten Kritik an semantischen Ansätzen zur Interoperabilität wurde Forschungsfrage 1 formuliert. Wenn die Kritik an der Semantik zur Herstellung der Interoperabilität, besonders in Bezug auf Fertigungssysteme, gerechtfertigt ist, stellt sich die Frage, ob die pragmatische Interoperabilität, die von den Kritikern der Semantik vorgeschlagen wird, einen geeigneten Lösungsansatz darstellt (F_2) und wie ein solcher Lösungsansatz aussehen könnte (F_3).

Forschungsfrage 1 (F_1): *Inwiefern können die Behauptungen, dass Semantische Interoperabilität nicht ausreicht, unterstützt werden?*

Forschungsfrage 2 (F_2): *Wie kann das Wissen zur Anwendung von unterschiedlichen Geräten und/oder deren Komponenten so modelliert werden, dass dieses in einem komplexen Gesamtsystem verknüpft werden kann?*

Forschungsfrage 3 (F_3): *Wie können aus einem solchen Modell eines komplexen Systems Steuerungsprozesse für das Gesamtsystem abgeleitet werden?*

4.2 Hypothesen und Methoden

Forschungsfrage 1 bezieht sich auf die Anwendbarkeit semantischer Interoperabilität zur Lösung von Interoperabilitätsproblemen mit spezifischem Bezug zu Fertigungssystemen. Da es sich bei diesen um heterogene, komplexe Systeme handelt, wurde Hypothese 0 unter dieser Prämisse definiert und muss in Form einer Nullhypothese falsifiziert werden. Um die Nullhypothese H_0 zu testen wird zunächst mit einem Gedankenexperiment der Zusammenhang zwischen dem menschlichen und dem maschinellen Verständnis der Semiotik und der Interoperabilität geklärt und daraus dann die Anwendbarkeit für die semantische Interoperabilität für einfache bis komplexe Systeme abgeleitet. Wenn die Nullhypothese H_0 falsifiziert werden kann, muss im Gegenzug die Alternativhypothese H_1 gültig sein.

Hypothese 0 (H_0): *Semantische Interoperabilität reicht aus, wenn damit ein heterogenes, komplexes Systems so beschrieben werden kann, dass daraus eine Steuerungslogik abgeleitet werden kann.*

Hypothese 1 (H_1): *Semantische Interoperabilität reicht nicht aus, um damit ein heterogenes, komplexes Systems so zu beschreiben, dass daraus eine Steuerungslogik abgeleitet werden kann.*

Methode 1 (M_1): *Gedankenexperiment zur Semantik und Pragmatik und Ableitung der Anwendbarkeit auf komplexe Systeme.*

Mit Forschungsfrage 2 soll eine geeignete Methode zur Modellierung des in den einzelnen Geräten und Komponenten implizit vorhandenen Wissens (der jeweiligen Herstellerfirma) gefunden werden. Mit dieser Methode sollen die Geräte und deren Wissen in der letztendlichen Anwendung im Gesamtsystem verknüpft werden können. Mit Hypothese 2 wird ein Ansatz auf Basis der Pragmatik gewählt, der in Unterabschnitt 5.2.3 näher beschrieben und in Kapitel 6 durch die Modellierung einer Werkzeugmaschine für eine Fertigungszelle testweise umgesetzt wird. Wenn sich diese Modellierungsmethode für ein solches komplexes System eignet, ist die Wahrscheinlichkeit hoch, dass

auch komplexe Fertigungssysteme allgemein auf dieses Weise modelliert werden können. Es wird folglich ein induktives Vorgehen zur Untersuchung der Forschungsfrage angewendet.

Hypothese 2 (H_2): *Das Wissen zur Anwendung einer Werkzeugmaschine und deren Komponenten, im Kontext einer roboterisierten Fertigungszelle, kann durch deren Pragmatik, in Form von Zustandsdiagrammen und Übergangsbedingungen, beschrieben, und durch deren Semantik mit den Komponenten eines Handhabungsgerätes verknüpft werden.*

Methode 2 (M_2): *Induktives Vorgehen durch Modellierung einer Werkzeugmaschine für den Einsatz in einer roboterisierten Fertigungszelle.*

Die Modellierung des Wissens bringt letztendlich nur einen Vorteil zur Unterstützung einer agilen Automatisierung und Inbetriebnahme, wenn daraus zumindest teilautomatisiert Erkenntnisse gewonnen werden können, die nicht naheliegend sind. Mit Forschungsfrage 3 stellt sich in diesem Kontext die Frage nach der Ableitung von Steuerungsprozessen für die gesamte Fertigungszelle. Mit Hypothese 3 wird ein dafür möglicher Lösungsansatz formuliert, der dann in Unterabschnitt 5.2.4 vollständig definiert wird. Zur Überprüfung der Hypothese wird wieder ein induktives Vorgehen gewählt, bei dem das in Kapitel 6 entwickelte Modell einer Werkzeugmaschine als Grundlage dient, um daraus in Kapitel 7 einzelne Automatisierungsprozesse im Kontext einer Fertigungszelle abzuleiten.

Hypothese 3 (H_3): *Ein Zusammenführen der orthogonalen und hierarchischen Zustandsautomaten der Fertigungszelle in einen einzelnen, flachen Graphen ermöglicht es, mit anschließender Graphsuche eine Sequenz von Automatisierungsschritten zum Erreichen von bestimmten Zielen abzuleiten.*

Methode 3 (M_3): *Induktives Vorgehen durch Ableitung von Automatisierungsprozessen im Kontext der Fertigungszelle aus dem Modell der Werkzeugmaschine.*

4.3 Interoperabilität von technischen Systemen

Wird angenommen, dass ein technisches System aus mehreren einzelnen Komponenten, wie Geräten, Anwendungen und Teilsystemen, genau dann funktioniert, wenn alle Aspekte der Interoperabilität zwischen den Systemkomponenten erfüllt sind. So ist dabei für die Funktion des Systems zunächst unerheblich, ob die Fähigkeiten zur

... Interoperabilität	**senkt die Inbetriebnahmeaufwände auf Grund ...**
Konzeptuelle	unterschiedlicher Anwendungskonzepte für ähnliche Systeme
Dynamische	der Kombination unterschiedlicher Services oder der Veränderung des Systems
Pragmatische	unterschiedlicher Auswirkungen von verwendeten Services in Abhängigkeit vom Kontext
Semantische	unterschiedlicher und unterschiedlich verwendeter Begriffe und deren Bedeutung in einer Domäne
Syntaktische	unterschiedlicher Protokolle, Datenstrukturen und Datentypen
Technische	unterschiedlicher physischer Übertragungsmedien und Hardwareschnittstellen

Abbildung 4.1: Gründe für Inbetriebnahmeaufwände.

Erfüllung einzelner Aspekte von den Komponenten selbst bereitgestellt oder nachträglich hinzugefügt werden. Um aber von Interoperabilität reden zu können, müssen diese Fähigkeiten bereits vorab im System vorhanden sein und sich miteinander organisieren können. Durch die unterschiedlichen Aspekte der Interoperabilität werden nun generell Aufwände bei der Inbetriebnahme reduziert, welche durch die Vielfalt unterschiedlicher Systeme entstehen (Abbildung 4.1), da diese Funktionalitäten nicht erst noch zusätzlich hinzugefügt werden müssen. Um eine vollständige Interoperabilität im Sinne von P&P zu erreichen, müssen nun alle Aspekte der Interoperabilität im Gesamtsystem vorhanden sein. Diese Arbeit definiert als **Ziel der Interoperabilität**, von abstrakten konzeptuellen Modellen und Zielvorgaben des Systems ausgehend funktional unterschiedliche Komponenten zur Erfüllung von Zielvorgaben nutzen zu können und dabei Aufwände zur Konfiguration oder Programmierung zu minimieren.

Die Aspekte der **technischen und syntaktischen Interoperabilität** führen dabei noch bis heute zu Aufwänden bei der Inbetriebnahme von Anlagen, die auf unterschiedliche Übertragungsmedien, Schnittstellen und Protokolle zurückzuführen sind. Die Industrie hat aber bereits erkannt, dass sich diese Probleme durch einheitliche und unabhängige Technologien zur Datenübertragung, wie Ethernet und OPC UA lösen lassen. Standardisierungsbemühungen in diesem Bereich sind daher besonders in den letzten Jahren schon weit fortgeschritten.

Durch eine einheitliche **Semantik** sollen Anlagen, Maschinen und deren Komponenten nun so beschrieben werden können, dass Anwendungen mit unterschiedlichen

Geräten und Diensten einheitlich kommunizieren können und dadurch der Aufwand sinkt, die Beschreibungen unterschiedlicher Schnittstellen zu verstehen. Dabei erfolgt letztendlich eine Standardisierung des verwendeten Vokabulars und der mit den einzelnen Begriffen verknüpften Bedeutung. Die ersten Bemühungen für solche Standardisierungen, hat es mit dem Semantic Web [82] gegeben, vor dessen Hintergrund seitdem eine Vielzahl an Ontologien im Internet veröffentlicht wurden. In den letzten Jahren haben sich auch um den Industriestandard OPC UA viele Arbeitsgruppen [27] gebildet, um Begleitspezifikationen zu definieren. Viele sehen darin die Lösung, um die Interoperabilität zwischen Geräten unterschiedlicher Herstellerfirmen zu ermöglichen, vernachlässigen dabei aber, dass die Semantik es zwar erlaubt, Komponenten und deren Funktionen eindeutig zu erkennen, aber nicht die Auswirkungen einer Funktion im Kontext einer Maschine, eines Systems oder Prozesses.

Auch bei der **konzeptuellen Interoperabilität** gibt es Entwicklungen. Mit Skill-based Engineering können Prozesse mittels abstrakter Skills auf einem konzeptuellen Niveau beschrieben werden. Auch mit Basisontologien kann ein Sachverhalt möglichst breit beschrieben werden, um eine möglichst hohe Wiederverwendbarkeit zu besitzen. Konzeptuelle Interoperabilität liefert eine einfache und breite Standardisierbarkeit, ohne sich auf Details der Implementierung einzuschränken.

4.4 Semantik und Pragmatik als Grundlage der Interoperabilität

Die Frage, ob Semantik allein ausreicht, um eine durchgängige Interoperabilität zu ermöglichen, ist sicherlich vom Anwendungskontext abhängig. Pfrommer et al. [45] argumentieren, dass semantische Modellierung für P&P (und damit vollständige Interoperabilität über alle Ebenen) nicht ausreicht, da für die Anwendung sehr detaillierte Beschreibungen notwendig sind, aber mit zunehmendem Detaillierungsgrad auch die allgemeine Verwendbarkeit der Ontologie sinkt und deren Anwendungsbereich damit stark eingeschränkt wird. Dies steht im Widerspruch zum Ziel der Semantik, in einer bestimmten Domäne in der Industrie eine möglichst breite Anwendbarkeit zu besitzen. Dagegen könnte man nun argumentieren, dass Ontologien genau dafür das Konzept von Taxonomien bzw. Vererbung mitbringen, um mit einer Basisontologie eine breite Anwendbarkeit zu gewährleisten und durch Spezialisierungen dieser Ontologie einzelne Sonderlösungen zu integrieren, die weiterhin über die Basisontologie zugeordnet werden können. Jedoch wird die Verwendbarkeit von Spezialisierungen von Konzepten in Ontologien auch von der Verwendbarkeit der Konzepte in der Basisontologie bestimmt. Alle zusätzlichen Konzepte und Beziehungen zwischen diesen, die nicht von der Basi-

sontologie abstammen, sind für die Anwendungsseite unbekannt und müssen wiederum neu interpretiert und programmiert werden.

Um Hypothese 0 zu überprüfen wird zuerst mit einem Gedankenexperiment der Zusammenhang zwischen dem menschlichen und dem maschinellen Verständnis der Semiotik und der Interoperabilität geklärt und daraus dann die Anwendbarkeit für die semantische Interoperabilität für einfache bis komplexe Systeme abgeleitet.

4.4.1 Gedankenexperiment zur Semiotik

Aus Sicht der Semiotik benötigt der Mensch zum Erkennen und Verstehen von Zeichen Syntax, Semantik und Pragmatik. Durch die Syntax erkennt er Symbole und Zeichen, wie das Telefon und die Kamera auf der linken Seite von Abbildung 4.2 und kann diese mittels Semantik den ihm bekannten Konzepten eines "Telefons" und einer "Kamera" zuordnen. Der Mensch kann beide ihm bekannten Geräte verwenden, weil er die Erfahrung bzw. die Pragmatik zur Verwendung der Geräte mitbringt. *Übertragen auf eine Fertigungszelle, benötigt diese ebenso das Wissen zum Steuern der Maschinen in dieser Zelle. Die Standardisierung der Semantik erlaubt es der Steuerung die Maschinen eindeutig zu erkennen und zu verwenden, für die die Steuerung, die Semantik und Pragmatik kennt.* Nun wird als neues Gerät das Smartphone eingeführt, das semantisch gesehen eine Spezialisierung des Telefons und der Kamera darstellt, die gleichen Funktionen zum Telefonieren und Fotografieren mitbringt, aber dem Menschen bis dahin noch nicht bekannt ist. Die zwar semantisch gesehen gleichen Funktionen des

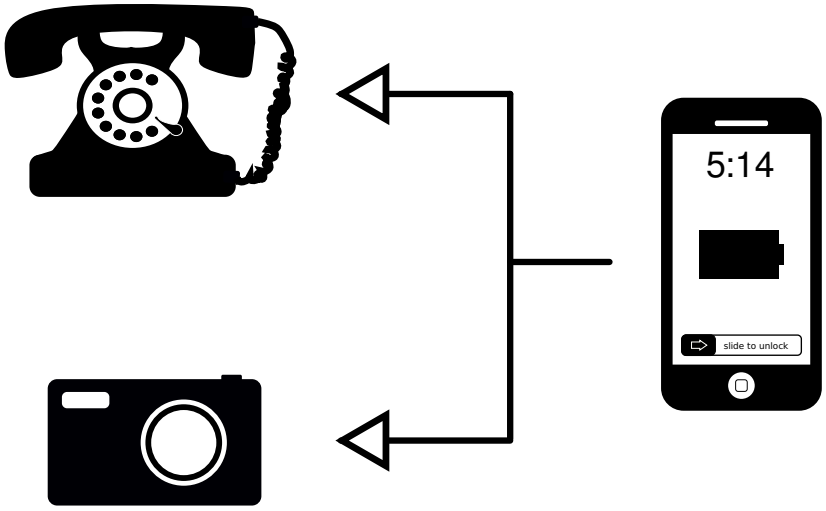


Abbildung 4.2: Gedankenexperiment zur Interoperabilität durch Semiotik.

Smartphones sind jedoch nicht auf die gleiche Art zu verwenden, wie deren Generalisierungen, sondern sind von mehreren Bedingungen, wie u. a. dem Entsperren des Gerätes und dem Starten der jeweiligen Anwendung abhängig. Ist dem Menschen nun zwar die Semantik bekannt, aber nicht die Pragmatik, also die Ursache-Wirkung Beziehungen, die das implizite Wissen zur Anwendung des Smartphones liefern, kann der Mensch das Smartphone nicht verwenden. *Analog für Fertigungssysteme gilt, wenn alleine Semantik bei der Beschreibung der Maschinen verwendet werden soll, dass neue Maschinen mit Sonderfunktionen nicht verwendet werden können, so lange die Pragmatik dazu nicht auch standardisiert und dann in der Steuerungssoftware hinterlegt wird.*

4.4.2 Zusammenspiel von Syntax, Semantik und Pragmatik

Es gibt aber auch eine weitere Möglichkeit, die sich der Mensch normalerweise zu Nutzen macht. Pragmatik muss nicht mit der Semantik standardisiert werden, sondern kann gerätespezifisch sein und vom Gerät beschrieben werden. In Abbildung 4.2 bringt das Smartphone die Pragmatik zum Entsperren bereits selbst so mit, dass der Mensch die Verwendung des Geräts zum Entsperren interpretieren kann. Die Semiotik liefert dafür die Grundlage mit Syntax (Symbole für Pfeil und Text), Semantik (Erkennen als Pfeil und Bedeutung des Texts) und Pragmatik (Intention des Pfeils nach rechts und des Texts zu einer Ursache-Wirkung Beziehung zum Entsperren). Analog dazu werden bei Maschinen und Anlagen Handbücher zu Betrieb und Bedienung durch die Herstellerfirma mitgeliefert, um den Menschen bei der Inbetriebnahme und Anwendung zu unterstützen. Folglich muss die darin enthaltene Pragmatik, in einer für eine Zellensteuerung verständlichen Weise, digitalisiert und durch das spezifische Gerät bereitgestellt werden, um eine Interoperabilität auch für neue Geräte zu ermöglichen. Die Bereitstellung der Semantik alleine reicht hierfür nicht aus.

4.4.3 Semantische Interoperabilität komplexer Systeme

Durch das Gedankenexperiment ist nun die Rolle der Syntax, Semantik und Pragmatik im Kontext von Fertigungssystemen definiert und geklärt, wieso Semantik nicht ausreicht, um auf die Funktionsweise neuer und spezialisierter Geräte oder Anwendungen zu schließen, so lange diese nicht identisch ist. Die semantische Interoperabilität zielt aber auch nicht darauf ab, neue Geräte derartig verwenden zu können, sondern eine einheitliche Semantik für diese zu schaffen, um deren Fähigkeiten eindeutig zu identifizieren. Jede Anwendung, die diese Semantik kennt, weiß auch mit dieser umzugehen.

Wenn man folglich alleine auf die semantische Interoperabilität setzt, so steckt die zugehörige Pragmatik in der Anwendung, welche die Regeln, Beziehungen und Prozesse kennt und anwendet. Diese Pragmatik muss bezogen auf die Semantik standardisiert sein. Für eine Fertigungszelle setzt die semantische Beschreibung der Komponenten alleine also voraus, dass die Pragmatik für diese semantische Beschreibung standardisiert und in der Zellensteuerung programmiert ist. *Um Hypothese 0 zu überprüfen und damit zu klären, ob diese Art der Beschreibung von Systemen zielführend ist, muss die Art der Systeme und der damit verbundene Aufwand zur Standardisierung der Pragmatik auf Basis der semantischen Beschreibung der Systeme betrachtet werden.* Dabei muss einerseits zwischen homogenen und heterogenen Systemen unterschieden werden, also zwischen Systemen, die immer aus den gleichen Komponenten bestehen, und Systemen, die je nach Anwendung unterschiedliche Komponenten besitzen. Außerdem spielt der Komplexitätsgrad eine Rolle. Einfache bis sehr umfangreiche Systeme können dabei zwar wenige bis sehr viele Komponenten besitzen, die Abhängigkeiten der Komponenten untereinander ist aber vernachlässigbar. Komplexe Systeme dagegen besitzen Komponenten, die eine Vielzahl von Abhängigkeiten untereinander besitzen. Laut Van Brussel [122] kann bei einfachen bis sehr umfangreichen Systemen das Verhalten dieser Systeme noch vorausgesehen werden, da die Abhängigkeiten zwischen Systemkomponenten überschaubar sind. Bei komplexen Systemen, wie Fertigungssystemen, kann das Verhalten des Gesamtsystems aber nicht mehr aus dem Wissen über die Einzelkomponenten und Subsysteme abgeleitet werden. In Anlehnung an die Vorhersehbarkeit des Verhaltens von einfachen, schwierigen und komplexen Systemen nach Van Brussel [122] gilt damit für:

1. **homogene, einfache bis komplexe Systeme**, dass die Beschreibung mit Semantik durch die immer gleiche Kombination von sehr ähnlichen Komponenten ausreicht, damit die gleiche Steuerungslogik wiederverwendet werden kann. Der Aufwand, die Steuerungslogik zu erstellen, ist selbst für komplexe Systeme überschaubar, da die gleiche Logik für homogene Systeme immer wiederverwendet werden kann.

2. **heterogene, einfache bis sehr umfangreiche Systeme**, dass die Beschreibung der einzelnen Komponenten und deren Verknüpfung mit Semantik ausreicht, da das Verhalten aus dem Wissen der Steuerungslogik (zu Pragmatik und Dynamik in Kontext der Anwendung) inferiert werden kann und die Anzahl der unterschiedlichen Regeln dabei überschaubar bleibt. Selbst wenn weitere Komponenten zum System dazukommen, können diese mit den in der Steuerung hinterlegten Regeln verwendet werden, da keine Auswirkungen auf andere Komponenten, wie in einem komplexen System, bestehen.

3. **heterogene, komplexe Systeme**, dass durch die Vielzahl an möglichen Kom-

binationen von und Abhängigkeiten zwischen unterschiedlichen Systemkomponenten die Anzahl an Regeln, die in der Steuerungslogik von vornherein bedacht werden müssen, enorm steigt. Mit steigender Komplexität der Systeme steigen dabei auch die Möglichkeiten an unterschiedlichen Kombinationen exponentiell. Die dabei für die Steuerung notwendigen Regeln können dabei nicht mehr im Vorhinein abgeschätzt werden und widersprechen damit Hypothese 0. Daher ist für diesen Fall die Beschreibung des Systems allein mit Semantik unzureichend.

Grundsätzlich lässt sich sagen, dass Semantik bereits einen großen Vorteil durch die Identifizierbarkeit und einheitliche Beschreibung von Komponenten liefert. Da die Welt jedoch heterogen ist und sich Maschinen unterschiedlicher Herstellerfirmen nicht wegen ihrer Gleichartigkeit, sondern ihrer oft proprietären oder maschinenspezifischen Eigenschaften durchsetzen, hilft eine einheitliche übergeordnete Semantik zur Integration heterogener Anlagenkomponenten in ein komplexes System, wie ein Fertigungssystem, alleine nicht weiter. Hypothese 0 konnte damit falsifiziert werden. Semantische Interoperabilität alleine ist damit nicht nur, wie im vorhergehenden Gedankenexperiment gezeigt, für die Verwendung neuer spezialisierter Geräte ungeeignet, sondern auch für die Verwendung in heterogenen, komplexen Systemen. Das bedeutet jedoch nicht, dass die Pragmatik alleine den Anspruch hat, diese Problematik zu lösen, sondern dass zur semantischen Beschreibung von Systemen, ebenfalls die pragmatische Beschreibung dieser notwendig ist.

Kapitel 5

Semiotische Interoperabilität agiler Fertigungszellen

In Abschnitt 4.4 wurde gezeigt, dass semantische Interoperabilität nicht ausreicht, um heterogene, komplexe Systeme, wie Fertigungszellen und deren Maschinen und Komponenten so zu beschreiben, dass daraus eine Steuerungslogik flexibel abgeleitet werden kann, da die dazu notwendigen Regeln im Vorhinein nicht vorhersehbar sind. Eine Lösung dafür könnte die pragmatische Interoperabilität und die damit verbundene Beschreibung des eigenen Verhaltens und des Verhaltens in Kooperation mit anderen Geräten durch die Systemkomponenten selbst bieten. Dieses Kapitel beschreibt auf dieser Grundlage die Integration der pragmatischen Interoperabilität in die Komponenten von Fertigungszellen und deren Anwendung auf Zellenebene, unter Berücksichtigung aktueller Technologien im Bereich der IKT.

5.1 Verteiltes Wissen und Wissensträger

Grundsätzlich baut das Konzept auf der Annahme auf, dass Wissen, in einem System verteilt ist. Der Begriff Wissen basiert hier grundsätzlich auf der Unterabschnitt 3.1.1 nach Fuchs-Kittowski [54], und wird hier als der bekannte Zusammenhang bzw. die Abhängigkeiten zwischen Ursache und Wirkung von Aktionen in einem System betrachtet. In einem solchen verteilten System wird Wissen durch unterschiedliche Wissensträger (Menschen, Maschinen, Anwendungen, etc.), zur Verfügung gestellt. Dabei wird davon ausgegangen, dass die Kombination dieses Wissens ausreicht, um die notwendigen

Schritte abzuleiten, mit denen die Systemkomponenten zum Erreichen von bestimmten Systemzuständen gezielt manipuliert werden können. Die Kombination und Komposition dieses Wissens wird durch die Interoperabilität der Systemkomponenten ermöglicht. Blackler [123] unterscheidet die fünf Arten von Wissen *embrained, embodied, encultured, embedded* und *encoded*. Wissen kann demnach in einem Menschen, in Form von konzeptuellen Fähigkeiten und kognitiven Fähigkeiten, zum Verständnis komplexer Zusammenhänge, vorliegen (embrained) oder als handlungsbezogenes Wissen, das im praktischen Umgang mit physischen Dingen entsteht (embodied). Es kann ebenfalls in einem gemeinsamen Verständnis bzw. in gemeinsamen Regeln innerhalb von Organisationen vorliegen (encultured). Von eingebettetem (embedded) Wissen spricht man, wenn das Wissen in Form von Routinen, mit Regeln und Beziehungen vorliegt, und von kodiertem (encoded) Wissen, wenn dieses in Form von Zeichen und Symbolen (z. B. in Büchern) abgebildet ist.

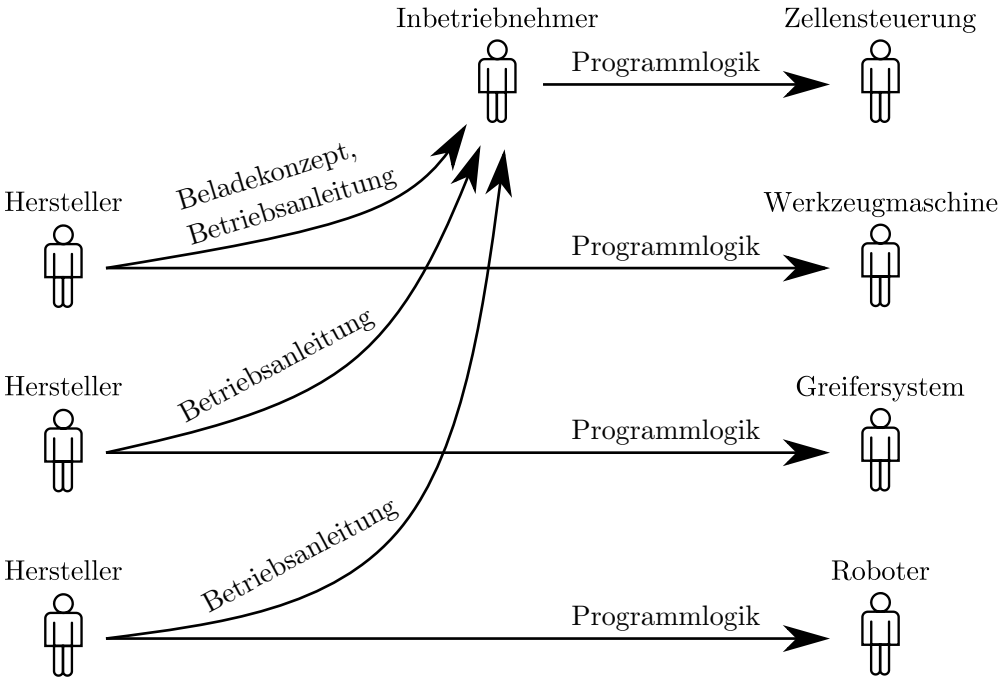


Abbildung 5.1: Wissen, Wissensträger und -transfer bei der klassischen Automatisierung einer Fertigungszelle.

5.1.1 Wissenstransfer bei der klassischen Automatisierung von Fertigungszellen

Wissensträger können in einem Fertigungssystem Objekte, wie Maschinen, Anwendungen, ein MES, aber auch Menschen sein. Die Form, in welcher das Wissen jeweils vorliegt, ist also sehr unterschiedlich. Das Wissen von Maschinen liegt, ebenso wie das Wissen von Anwendungen, einerseits explizit in dem in der Steuerung in Form von Regeln programmierten Verhalten vor (embedded). Andererseits liegt Wissen auch explizit in den zur Maschine gehörigen Betriebsanleitungen vor (encoded). Das Wissen von Menschen liegt stattdessen meist nur implizit in Form von Erfahrung vor (embrained und embodied). Bei der klassischen Automatisierung und Inbetriebnahme einer Fertigungszelle in Abbildung 5.1 bringt der Mensch einerseits Wissen bei der Programmierung der Zellensteuerung in Form von Programmierlogik (embedded) ein, das dieser teilweise aus seiner Erfahrung ableiten kann, andererseits aber auch Wissen zu maschinenspezifischen Eigenschaften, das aus der Betriebsanleitung stammt, die die Herstellerfirma der Maschine ursprünglich verfasst hat (encoded). Es findet also indirekt ein Wissenstransfer (Abbildung 5.2) von der Maschine (Sender), bzw. indirekt von der Herstellerfirma der Maschine, an die Zellensteuerung (Empfänger) statt. Dabei werden die spezifischen Fähigkeiten der Maschine und deren Randbedingungen, in

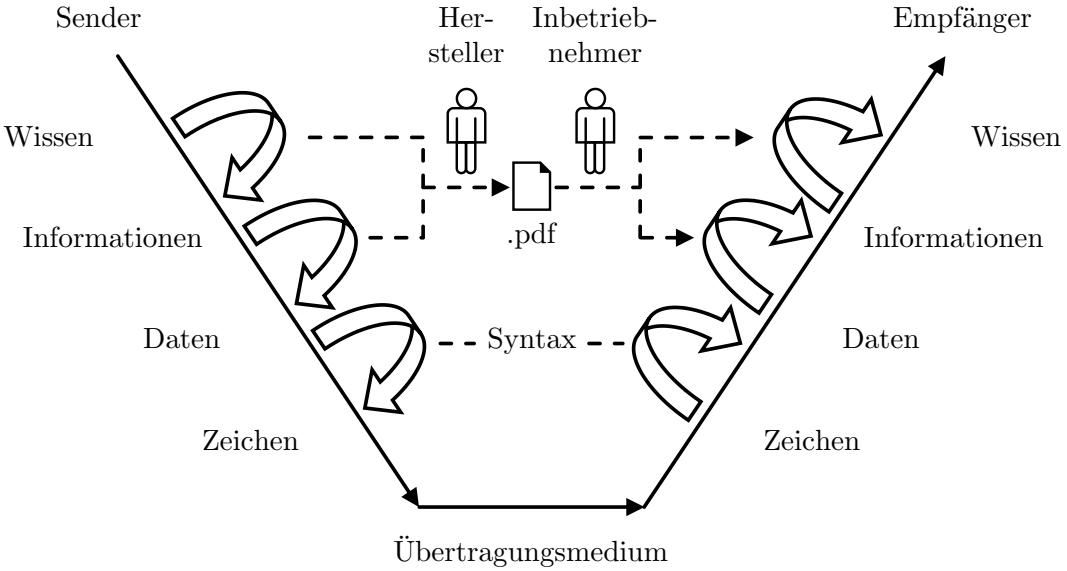


Abbildung 5.2: Wissenstransfer zur Zellensteuerung bei der klassischen Automatisierung (entprechend Abbildung 3.2).

ein von der Steuerung ausführbares Programm konvertiert. Direkt übertragen werden meist nur die Signale der Maschinen in einer standardisierten Form (Syntax). Semantik und Pragmatik werden dagegen durch die Herstellerfirma in der Betriebsanleitung dokumentiert, die dann wieder bei der Inbetriebnahme durch einen Menschen interpretiert und in die Zellensteuerung programmiert werden muss.

5.1.2 Wissenstransfer bei der agilen Automatisierung von Fertigungszellen

Zur agilen Automatisierung von Fertigungszellen müssen die Aufwände bei der Inbetriebnahme minimiert werden. Folglich muss das dazu notwendige Wissen, möglichst automatisiert bereitgestellt und in die Zellensteuerung integriert werden, um daraus eine Logik zur Steuerung der Zelle abzuleiten. In Abbildung 5.3 ist der dafür notwendige Wissenstransfer dargestellt. Im Gegensatz zur klassischen Automatisierung und dem damit verbundenen Wissenstransfer (Abbildung 5.1 und Abbildung 5.2), werden hierbei möglichst alle dafür notwendigen Informationen direkt von den Systemkomponenten (Maschinen und deren Subkomponenten) der Zellensteuerung bereitgestellt. Dabei enthält die von der Herstellerfirma in ihren Maschinen programmierte Logik indirekt auch deren Verhalten. Laut dem *process before data* Designprinzip für Anwendungen im Pragmatic Web nach Singh [99], [101], hat eine Person, die Prozesse und die dazugehörigen Daten modelliert, ein besseres Verständnis von diesen Daten als eine Person, welche die Daten ohne den Prozesskontext sieht. Folglich haben Herstellerfirmen auch ein besseres Verständnis über Verhalten und Anwendung ihrer Maschinen als die Personen, die diese letztendlich anwenden. Nach diesem Designprinzip ist es wichtig, dass Herstellerfirmen bereits die vorgesehene Anwendung der Maschine im Prozesskontext beschreiben und nicht nur die dazugehörigen Daten zu Struktur und Aufbau. Dieses Wissen muss von der Herstellerfirma anstatt in einem Handbuch zur Inbetriebnahme und Konfiguration direkt in der Maschine hinterlegt werden, so dass diese ihre Anwendung im Prozesskontext selbst beschreibt. Es enthält das Verhalten der Maschine, basierend auf den logischen, von der Herstellerfirma programmierten Regeln und Beziehungen zwischen den Subkomponenten der Maschine selbst und den Subkomponenten anderer Maschinen.

Bereits in Bezug auf die klassische Automatisierung in Abbildung 5.1 müssen durch die Herstellerfirma einer Maschine oft schon Konzepte zur Integration dieser, im Kontext einer Verschaltung mit anderen Komponenten in einem System, mitbedacht werden. Im Falle von Werkzeugmaschinen, die für eine Automatisierung und damit die Integration in Fertigungszellen bzw. Fertigungssysteme ausgelegt sind, beinhaltet dies traditionell u. a. Konzepte zur Werkstückhandhabung (Be- und Entladen der Werk-

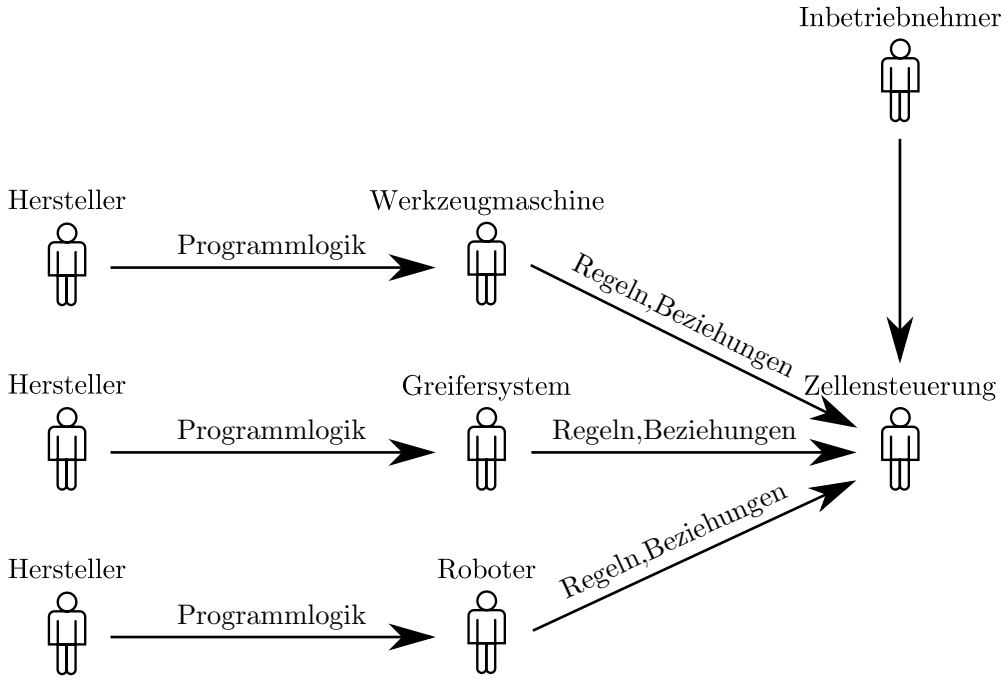


Abbildung 5.3: Wissen, Wissensträger und -transfer bei der agilen Automatisierung einer Fertigungszelle.

zeugmaschine) in Kombination mit einer Handhabungseinrichtung wie einem Roboter-Manipulator, und die dazu notwendige Schnittstellenprogrammierung in der Maschine selbst durch die Herstellerfirma. Teilweise werden in neueren Maschinen bereits flexible Automatisierungskonzepte wie die VDMA 34180 [124] angewandt, mit der proprietäre Konzepte zum Zweck einer einheitlichen Lösung ersetzt werden sollen. Eine solche Lösung erlaubt es unterschiedliche Ansätze verschiedener Herstellerfirmen auf wenige unterschiedliche Anwendungsszenarien und dazugehörige Schnittstellen zu reduzieren, und dadurch die Integration zu vereinfachen. Solche Konzepte könnten aber auch eine Basis für die Integration neuer Funktionen bilden, wenn diese einerseits ein grundsätzlich einheitliches Verständnis der Domäne ermöglichen, aber tiefergreifend auch proprietäres und maschinenspezifisches Wissen (und dessen Semantik und Pragmatik) verarbeiten können.

Das Konzept für die gesamte Fertigungszelle wird zwar meist durch einen Menschen für einen speziellen Anwendungsfall geplant, jedoch müssen dabei die Systemkomponenten (Werkzeugmaschine und Roboter) selbst bereits auf konzeptueller Ebene interoperabel sein. Die tatsächliche Anwendung wird also letztendlich vom Menschen vorgegeben, die Systemkomponenten liefern aber oft bereits selbst Konzepte für die Integration externer Systeme mit, die besonders für die Integration proprietärer Funk-

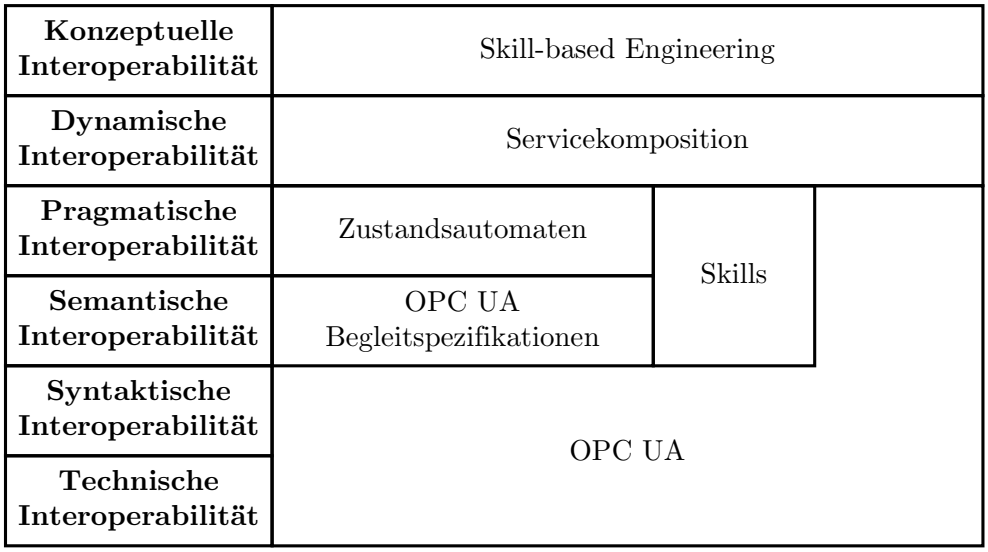


Abbildung 5.4: Bausteine der semiotischen Interoperabilität agiler Fertigungszellen, basierend auf dem Levels of Conceptual Interoperability Model (LCIM).

tionen unerlässlich sind. Die Beschreibung dieser Konzepte und der Interaktion der Systemkomponenten ist Teil des proprietären Wissens, das für die Planung des Gesamtsystems auf konzeptioneller Ebene und die Integration der Komponente im Sinne einer agilen Automatisierung und Inbetriebnahme notwendig sind. Auch dieses Wissen wird bereits beim Erstellen der Programmlogik der jeweiligen Komponente durch die Herstellerfirma indirekt in der Komponente kodiert und kann damit ebenso durch Regeln und Beziehungen beschrieben werden, die jedoch mit konzeptuellen externen Komponenten verknüpft sind.

5.2 Wissensaustausch durch Interoperabilität

Damit die automatische logische Verschaltung und Komposition der Einzelkomponenten in einem System ermöglicht wird, muss das dazu notwendige Wissen von den Komponenten in einem geeigneten Format bereitgestellt werden, um dadurch eine ganzheitliche Interoperabilität zu realisieren. Hierfür eignen sich verschiedene Technologien (Abbildung 5.4), die für die unterschiedlichen Arten der Interoperabilität genutzt werden können.

5.2.1 Technische und syntaktische Interoperabilität

Auf der Ebene der technischen und syntaktischen Interoperabilität existieren eine Vielzahl unterschiedlicher Übertragungsmedien und Protokolle. Herkömmliche Fertigungsmaschinen und Handhabungsgeräte, wie man sie großteils im Brownfield bei Unternehmen antrifft, setzen dabei meist auf einen binären Signalaustausch von Speicherprogrammierbaren Steuerungen (SPS) über elektrische Signale oder einen Feldbus. Mit der steigenden Akzeptanz von OPC UA, MTConnect und anderen Ethernet-basierten Kommunikationsstandards in der Industrie, und der damit einhergehenden Unterstützung dieser Standards durch unterschiedliche Herstellerfirmen, ist damit zu rechnen, dass diese Ebene in den kommenden Jahren kein Problem mehr darstellen wird. Diese Prognose bezieht sich zwar hauptsächlich auf Greenfield Anlagen, im Brownfield wird aber ebenfalls nach Lösungen zum Hochrüsten vorhandener Anlagen gesucht.

Im Rahmen dieser Arbeit dient OPC UA auf der Ebene der technischen und syntaktischen Interoperabilität (Abbildung 5.4) zur Kommunikation der Systemkomponenten und über die graph-basierte Syntax von OPC UA auch der Modellierung der Komponenten auf den folgenden Ebenen der Interoperabilität.

5.2.2 Semantische Interoperabilität

Die semantische Interoperabilität ist eine der Kernherausforderungen von I4.0, IoT und IIoT, und hat sich damit in den vergangenen Jahren nicht nur in der Forschung, sondern auch in der Industrie als gemeinsames Ziel etabliert. Im Bereich von Fertigungssystemen wurde in dieser Zeit mit der Entwicklung von mehreren OPC UA Begleitspezifikationen begonnen. Von den mittlerweile 54 Arbeitsgruppen für OPC UA [27] sind derzeit allein 20 unter *Manufacturing Devices, Robots, Machines & Machine Tools* gelistet. Zu den in diesen Gruppen entwickelten Informationsmodellen zählen abstrakte Modelle wie *OPC UA for Machinery*, die generelle Maschinenkomponenten beschreiben, aber auch spezifischere Modelle, die Maschinentypen und deren Komponenten selbst definieren. Für **Werkzeugmaschinen** wurden einerseits *OPC UA for CNC Systems* und später *universal machine technology interface (umati)* (früher: *universal machine tool interface*) entwickelt. Auch schon vor OPC UA haben sich mit dem Kommunikationsstandard MTConnect Informationsmodelle [125] für Werkzeugmaschinen und deren Struktur [126], Assets wie Werkzeuge zur Zerspanung [127], [128] und letztendlich auch für Schnittstellen zu anderen Komponenten [129] etabliert, die auch heutzutage noch weiterentwickelt werden. MTConnect besitzt mit *OPC UA for MTConnect* ebenfalls eine Begleitspezifikation, wodurch insgesamt drei unterschiedliche

Standards für die Semantik von Werkzeugmaschinen in OPC UA von unterschiedlichen Organisationen vorliegen. Für **Industrieroboter** existiert die Begleitspezifikation *Robotics* und für deren Werkzeuge, wie z. B. Greifer, befindet sich derzeit *End-of-arm Tools* in Entwicklung.

Daneben kommt in der Forschung auch das Thema semantisch standardisierter Skills auf, um generische Fähigkeiten beschreiben zu können und damit die Grundlage für die Steuerung von Anlagen zu schaffen, auch wenn hier noch keine offiziellen Standards existieren. Mit Skills lassen sich Maschinen eher aus Sicht ihrer Fähigkeiten als aus Sicht ihrer Struktur und Komponenten beschreiben. Diese erlauben damit neben ihrer semantischen, auch eine pragmatische Sicht auf die damit beschriebenen Geräte.

Da OPC UA durch standardisierbare und leicht austauschbare Modelle zur Beschreibung von Geräten und Skills gut geeignet ist, und vor allem auf die Verknüpfung der modellierten Geräte mittels Semantik abzielt, eignet sich OPC UA auch besonders zur Anwendung auf Ebene der Semantik im Rahmen der in dieser Arbeit vorgeschlagenen Methodik (Abbildung 5.4). Dies erlaubt einerseits die Wiederverwendung bereits semantisch standardisierter Begleitspezifikationen, aber auch die Definition eigener Semantik im Bezug auf nicht standardisierte Skills.

5.2.3 Pragmatische Interoperabilität

Davon ausgehend, dass semantische Interoperabilität bereits in Forschung und Industrie umfangreich behandelt wurde, ist dagegen die pragmatische Interoperabilität in dieser Tiefe noch weniger erforscht. Letztere ist im Zusammenhang mit der dynamischen Interoperabilität der eigentliche Hauptansatzpunkt dieser Arbeit. Pragmatische Interoperabilität dient hierbei der Modellierung der Funktionen und des Verhaltens der einzelnen Maschinen in der Fertigungszelle mittels Zustandsautomaten (Abbildung 5.4). Sie beinhaltet das proprietäre Wissen der jeweiligen Systemkomponente der Fertigungszelle in Form von Zuständen und dabei möglichen Zustandsübergängen. Eine Maschine ist darin nicht als einzelner Zustandsautomat, sondern als Set an Zustandsautomaten modelliert, die die einzelnen Subkomponenten der Maschine beschrieben, die für die Anwendung im Prozesskontext notwendig sind. Dadurch wird eine flexible Zusammenstellung von (möglicherweise sogar bereits standardisierten oder zumindest oft standardisierbaren) Subkomponenten ermöglicht. Das proprietäre Wissen wird traditionell bei der Zusammenstellung der Komponenten und der logischen Verschaltung dieser bei der Programmierung der Maschine in der jeweiligen Steuerung hinzugefügt. Im Kontext der pragmatischen Interoperabilität wird genau dieses Wissen ebenfalls in den Zustandsautomaten abgebildet, indem die maschinenspezifischen Bedingungen für

die Zustandsänderungen aus der Programmlogik auch den Zustandsübergängen hinterlegt werden. Dadurch werden die Abhängigkeiten der einzelnen Subkomponenten untereinander in dem Zustandsautomaten dokumentiert.

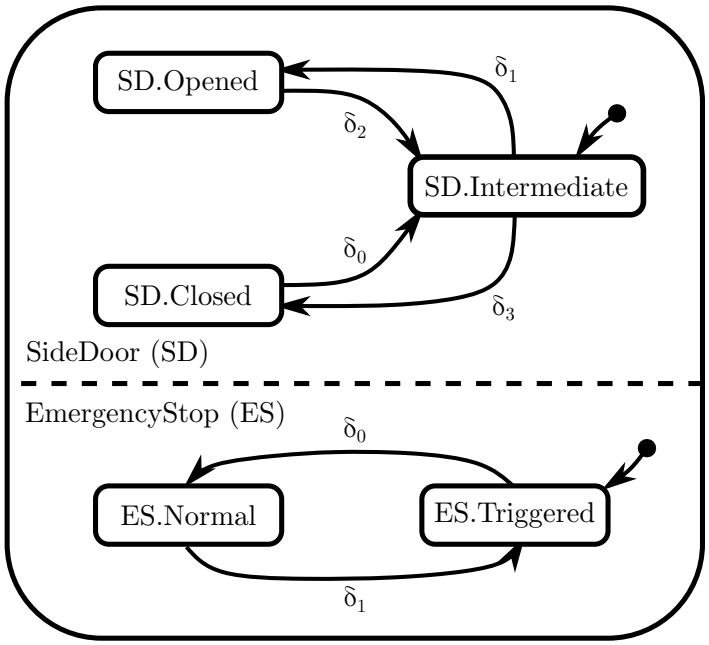


Abbildung 5.5: Beispiel für die pragmatische Beschreibung der Zustände, Zustandsübergänge und deren Abhängigkeiten untereinander.

Tabelle 5.1: Subzustände in den orthogonalen Zuständen SideDoor (SD) und EmergencyStop (ES).

$p_{0,0}$:	$SD.Intermediate$
$p_{0,1}$:	$SD.Opened$
$p_{0,1}$:	$SD.Closed$
$p_{1,0}$:	$ES.Triggered$
$p_{1,1}$:	$ES.Normal$

Als Beispiel hierfür wird die einfache Verschaltung von zwei zustandsabhängigen Subkomponenten, dem Not-Halt ES und der automatischen Beladetür SD einer Maschine, angegeben. In Abbildung 5.5 sind die beiden Subkomponenten $SD \Leftrightarrow \mathbb{P}_0$ und $ES \Leftrightarrow \mathbb{P}_1$ als zueinander orthogonale Zustände mit den dazugehörigen Zuständen $p_{n,m} \in \mathbb{P}_n$ (Tabelle 5.1) und den zugehörigen Zustandsübergängen $\delta_{n,j}$ (Tabelle 5.2)

modelliert. Dabei sind in den Zustandsübergängen der beiden Subkomponenten einerseits die Funktionen zum Auslösen der Zustandsübergänge als Trigger (*Open* und *Close*), und andererseits die Abhängigkeiten der Zustandsübergänge von internen Sensorsignalen (es_active) und orthogonalen Zuständen ($ES.Normal$, hier äquivalent zu $In('ES.Normal')$ in SCXML) als Bedingungen, modelliert.

Tabelle 5.2: Zustandsübergänge in den orthogonalen Zuständen SideDoor (SD) und EmergencyStop (ES).

$\delta_{0,0}:$	$SD.Closed$	$\times\ SD.Open\ [ES.Normal]$	$\rightarrow SD.Intermediate$
$\delta_{0,1}:$	$SD.Intermediate$	$\times\ SD.Open\ [ES.Normal]$	$\rightarrow SD.Opened$
$\delta_{0,2}:$	$SD.Opened$	$\times\ SD.Close\ [ES.Normal]$	$\rightarrow SD.Intermediate$
$\delta_{0,3}:$	$SD.Intermediate$	$\times\ SD.Close\ [ES.Normal]$	$\rightarrow SD.Closed$
$\delta_{1,0}:$	$ES.Triggered$	$\times\ [!es\_active]$	$\rightarrow ES.Normal$
$\delta_{1,1}:$	$ES.Normal$	$\times\ [es\_active]$	$\rightarrow ES.Triggered$

Obwohl hierbei die Zustandsdiagramme nach Harel [108], [109] und die Darstellung der Modelle mit SCXML die Basis bilden, unterscheidet sich das Ziel dieser pragmatischen Modellierung von der herkömmlichen Anwendung von Zustandsdiagrammen. Die Modellierung dient einzig und alleine dem Zweck, das Verhalten des Systems so zu beschreiben, dass daraus Vorschläge für Aktionen (Trigger) abgeleitet werden können, um gewisse Zustandsänderungen zu bewirken. Das Ziel es ist nicht, über diese Zustandsdiagramme eine Steuerungslogik des Systems selbst zu modellieren und auszuführen. Dazu wäre eine weitaus komplexere Zustandsmodellierung notwendig, die für diesen Zweck eine unnötige Verkomplizierung des Systems mit sich brächte. Beispielweise müsste dazu eine Überprüfung der Signalübergänge des tatsächlichen Systems erfolgen. Würde nämlich der Trigger $SD.Close$ in $\delta_{0,2}$ bei gültiger Bedingung $In('ES.Normal')$ (in Abbildung 5.5 bzw. Tabelle 5.2) ausgelöst, so würde in der Theorie ein sofortiger Zustandswechsel in den Zustand $SD.Intermediate$ erfolgen. In der Praxis müssten aber die internen Signale des Systems (Endschalter der automatischen Beladetür) überprüft werden und der Zustandsübergang erst dann erfolgen, wenn sich diese ändern. Mit dem Trigger dürfte dann lediglich eine Signalflanke geschickt werden, mit der ein Aktorsignal (Motor zum Schließen der Beladetür) aktiviert wird. Dieses müsste dann wieder als orthogonaler Zustand oder als Subzustand des aktuellen Systemzustandes modelliert werden.

Tabelle 5.4: Interne Signale und aggregierte Zustände von SideDoor (SD).

	sd_opened	**sd_closed**
SD.Opened	1	0
SD.Closed	0	1
SD.Intermediate	0	0

Bei der Modellierung des Systems zum Zweck der Darstellung der Pragmatik, muss bei den in der SPS vorhandenen Datenpunkten und Signalen, zwischen rein internen Signalen, Zuständen und Triggern unterschieden werden:

- **Interne Signale** entsprechen den Sensor- und Aktorsignalen (Not-Halt Signal *es_active* in Tabelle 5.2 oder Endschalter *sd_opened* und *sd_closed* in Tabelle 5.4), der zugrundeliegenden Steuerung (z. B. SPS). Diese können als zusätzliche Information in den endgültigen Zustandsautomaten dargestellt werden, wenn diese eine Relevanz zur Steuerung des Systems haben und nicht durch einen Zustand im Zustandsautomaten aggregiert dargestellt werden können.

- **Zustände** können eine Menge von internen Signalen und Bedingungen, zusammen mit zugehörigen internen Zustandsübergängen, aggregieren. Die Zustände *SD.Opened*, *SD.Closed* und *SD.Intermediate* in Tabelle 5.4 sind aus den Zustandssignalen der einzelnen Endschalter der Beladetür aggregiert. Ein internes boolesches Signal kann aber unter Umständen auch zwei Zustände ergeben (*ES.Normal*, wenn *es_active* den Wert 0 hat, ansonsten *ES.Triggered*). Zustände sind letztendlich dafür da, die zur Steuerung des Systems relevanten Zustände der Subkomponenten darzustellen und entsprechen aus diesem Grund nicht immer direkt den internen Signalen.

- **Trigger** stellen die für das steuernde System angebotenen atomaren Skills zur Verfügung. Das bedeutet auch, dass von extern nur diese Trigger aufgerufen werden können und die restlichen Elemente rein der Beschreibung des Systemzustandes dienen. Trigger müssen bei einem Aufruf interne Signale setzen, die bei positiver Überprüfung der internen Bedingungen einen Zustandsübergang auslösen.

```
1 FC 43 - Robot handling side door
2 Network 1: Handling interface - door open
3  AN   "es_active"     //emergencystop is inactive
4  AN   "nc_active"     //program is inactive
5  A    "sd_closed"     //side door is closed
6  A    "hk_active"     //handling key switch
7  A    "rb_open_door"  //robot external signal
8  =    "sd_open"       //set internal signal
```

Codeabschnitt 5.1: Interne Signale und Bedingungen im Code der Speicherprogrammierbaren Steuerung (SPS) [1].

Codeabschnitt 5.1 zeigt einen Ausschnitt aus der Programmlogik der internen Steuerung einer Werkzeugmaschine, das die Regeln für das Öffnen der Beladetür über eine herkömmliche Automatisierungsschnittstelle beschreibt. Dabei sind interne Signale in Form von Bedingungen (z. B. $A = AND$ und negiert $AN = AND\ NOT$, Zeilen 3-7) miteinander verknüpft. Sind die Bedingungen erfüllt wird hier ein Signal gesetzt (Zeile 8), mit dem ein Aktor zum Öffnen der Beladetür angesteuert wird. Diese Darstellung zeigt besonders die Abhängigkeiten zwischen den Zuständen unterschiedlicher Subkomponenten von Maschinen, aber auch den Zusammenhang zur Abbildung dieser mittels Zustandsautomaten. Der Zustand *Opened* kann hier nur aus dem Zustand *SD.Closed* (entsprechend *sd_closed*) aufgerufen werden, wenn weder ein Not-Halt Signal (*es_active*) anliegt, noch die numerische Steuerung (*nc_active*) aktiv ist und gleichzeitig der Schlüsselschalter für die automatisierte Handhabung (*hk_active*) aktiviert ist. Diese Abhängigkeiten werden im Zustandsübergang von *SD.Opened* nach *SD.Closed* als Bedingungen modelliert. Das Signal, das letztendlich über die Automatisierungsschnittstelle an die Maschine gesendet wird, um das Öffnen der Beladetür zu bewirken (*rb_open_door*), wird im Zustandsübergang als Trigger modelliert. Die zu Codeabschnitt 5.1 äquivalente Abbildung der Zustände und des dazugehörigen Zustandsübergangs ist in Abbildung 5.6 dargestellt.

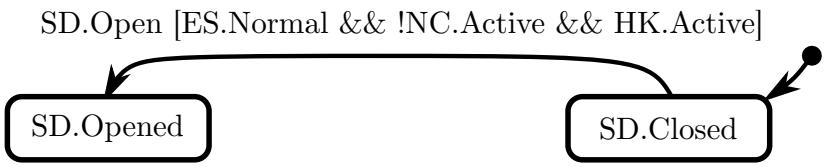


Abbildung 5.6: Zustandsautomat aus einem Programmabschnitt einer Speicherprogrammierbaren Steuerung (SPS).

Für die pragmatische Interoperabilität (Abbildung 5.4) bedeutet das, dass die einem SPS-Programm zugrundeliegende Logik auch in einem Zustandsautomaten pro Subkomponente vereinfacht dargestellt und die Beziehungen zwischen den Subkomponenten der Maschine in den Zustandsübergängen der Automaten hinterlegt werden

können. Dadurch können sich Maschinen und Anlagen selbst so beschreiben, dass Ursache, Wirkung und deren Abhängigkeiten für bestimmte Aktionen / Trigger identifizierbar sind. Externe Systeme können diese Informationen nutzen, um die Maschine zielgerichtet in einen geforderten Zielzustand zu bewegen. Um Trigger und Zustände eindeutig zu erkennen, ist Semantik notwendig. Es ist aber nicht für jeden Zustand und jede Aktion eine semantische Beschreibung nötig, denn letztendlich müssen, abhängig von der Anwendung, nur eine bestimmte Menge von Zielzuständen und Triggern eindeutig identifiziert werden können. Um die Verbindung zwischen der Semantik und Pragmatik herzustellen, eigenen sich einerseits die Zuweisung der Trigger und Zustandsübergänge zu semantisch standardisierten Skills, andererseits die Erstellung von Taxonomien für die Abstraktion von Zuständen. Für die Beschreibung der Zustandsautomaten zur Laufzeit eignet sich OPC UA, da es hierfür bereits eine standardisierte Darstellungsform für Zustandsautomaten gibt, die aber noch um Orthogonalität und Kommunikation, wie von Harel [108] beschrieben, erweitert werden muss. Da OPC UA auf einer graph-basierten Darstellungsform beruht, stellen solche Erweiterungen zur Erstellung von Spezialisierungen und zur Herstellung von Beziehungen zwischen Automaten aber keine Herausforderung dar.

5.2.4 Dynamische Interoperabilität

Da Fertigungssysteme aus mehreren Maschinen und Handhabungssystemen von verschiedenen Herstellerfirmen mit oft sehr unterschiedlichen Fähigkeiten bestehen, ergeben sich mit der Kombination dieser Maschinen komplexe und heterogene Systeme, deren Funktionen (und die Auswirkung dieser Funktionen), von den jeweiligen Zuständen der Maschinen im System abhängen. Die dynamische Interoperabilität muss daher die Komposition von Maschinen und deren Komponenten im Fertigungssystem in Abhängigkeit von veränderlichen Systemzuständen oder Konfigurationen beherrschen und daraus dynamisch die notwendigen Funktionen ableiten, mit denen die einzelnen Maschinen angesteuert werden müssen, um ein gemeinsames Ziel zu erreichen. Diese Komposition von Fähigkeiten funktioniert auf Basis von Semantik, Pragmatik und Konzepten.

Die von den einzelnen Maschinen bereitgestellte **Pragmatik** liefert die Grundlage für die dynamische Interoperabilität, um nicht nur die Interaktion mit der einzelnen Maschine zu beherrschen, sondern auch die Regeln zur Interaktion dieser Maschine mit externen Systemen zu erkennen und aus den Zuständen und Zustandsübergängen das dynamische Verhalten des Systems abhängig vom Zustand abzuleiten. Die **Semantik** erlaubt dabei die eindeutige Zuordnung bekannter Komponenten, Zustände und Zustandsübergänge. Dadurch können die Subkomponenten unterschiedlicher Maschinen

miteinander verknüpft werden, wenn zwischen diesen zur Steuerung relevante Beziehungen bestehen. Mit den abstrakten **Konzepten** der übergeordneten Ebene werden dabei generelle Verknüpfungen zwischen den Maschinenkomponenten hergestellt und zusätzliche Regeln und Bedingungen eingebracht.

Diese notwendige Verknüpfung zwischen den Maschinenkomponenten stellt dabei sicherlich eine der größeren Schwierigkeiten dar. Die grundsätzliche Kombination der Maschinen kommt dabei zwar aus dem Anlagenkonzept, detaillierte Steuerungsfunktionen sind aber bereits von der Pragmatik der einzelnen Maschinen in der Kombination mit anderen Maschinen abhängig. Betrachtet man die SPS Funktion zum Öffnen der Automatiktür in Codeabschnitt 5.1, so benötigt diese zum Öffnen nur ein externes Signal von einer Steuerung, das hier als Trigger modelliert wird. Betrachtet man dagegen die Funktion zum Schließen der Automatiktür, so benötigt die SPS ein zusätzliches Signal als Bedingung, das sicherstellt, dass das Handhabungssystem WorkpieceHandling (WH) den Maschinenraum verlassen hat ($WH.Outside$), also die Beladetür wieder freigegeben ist. $\delta_{2,0}$ und $\delta_{3,0}$ in Tabelle 5.2 besitzen also eine zusätzliche Bedingung $SD.Close$ [$ES.Normal$ && $WH.Outside$], die abhängig vom externen Handhabungssystem ist. Diese Bedingungen werden von der Maschine selbst mit ihrer Pragmatik, als Teil des spezifischen Wissens einer Herstellerfirma, beschrieben. Externe Systeme und Zustände können darin lediglich als zusätzliche Bedingungen auftreten, müssen als solche gekennzeichnet sein und sollten eine gewisse Semantik aufweisen, um einer tatsächlichen Komponente im Gesamtsystem zugeordnet werden zu können. Die konzeptuelle Ebene der Interoperabilität liefert dabei zusätzliche Informationen zur Verknüpfung der Maschinen, die der Herstellerfirma einer einzelnen Maschine nicht bekannt sein können, wie die genaue Anordnung der Systeme, wenn mehrere Handhabungssysteme und Fertigungsmaschinen vorhanden sind, oder ob die Maschinen mit zusätzlichen Komponenten, wie speziellen Endeffektorsystemen, ausgerüstet sind.

Die Kombination der semantischen, pragmatischen und konzeptuellen Ebene resultiert für die dynamische Interoperabilität in einem System aus logisch verknüpften Maschinen, mit jeweils auf deren Subkomponenten basierenden Zustandsautomaten, und einer Reihe an Bedingungen und Regeln, die die Verknüpfungen zwischen diesen beschreiben. Aus dieser Beschreibung des dynamischen Verhaltens des Gesamtsystems müssen letztendlich Aussagen getroffen werden können, die zum Erreichen von einem oder mehreren Zielzuständen führen. Ein möglicher Ansatz, Aussagen zu generieren, ist die graph-basierte Darstellungsform des Gesamtsystems aus Zustandsautomaten zu nutzen, um Pfade zwischen den Zuständen zu identifizieren. Algorithmen zur Wegfindung (*Pathfinding*) basieren jedoch meist auf einfachen uni- oder bidirektionalen Netzwerken, die weder orthogonale Netzwerke, noch zusätzliche Bedingungen mit parallelen Zuständen einschließen. Als Folge daraus müssen entweder neue Algorithmen dafür generiert, oder die orthogonalen Zustandsautomaten in eine entsprechende Form

gebracht werden. Die von Harel [108] eingeführten orthogonalen Zustände dienen zwar der Vereinfachung komplexer Systeme, sind aber mit herkömmlichen Wegfindungsalgorithmen nicht verwendbar. In Abbildung 3.10 ist der Zusammenhang zwischen einem flachen Zustandsautomaten mit impliziten orthogonalen Zuständen und seinem orthogonalen äquivalent dargestellt. Die Transformation zwischen diesen Darstellungsarten kann in beide Richtungen erfolgen. Folglich kann mit einer Rücktransformation nach bestimmten Regeln aus mehreren orthogonalen und hierarchischen Zuständen ein einzelner flacher Zustandsautomat und damit ein für die Wegfindung geeignetes Format erzeugt werden.

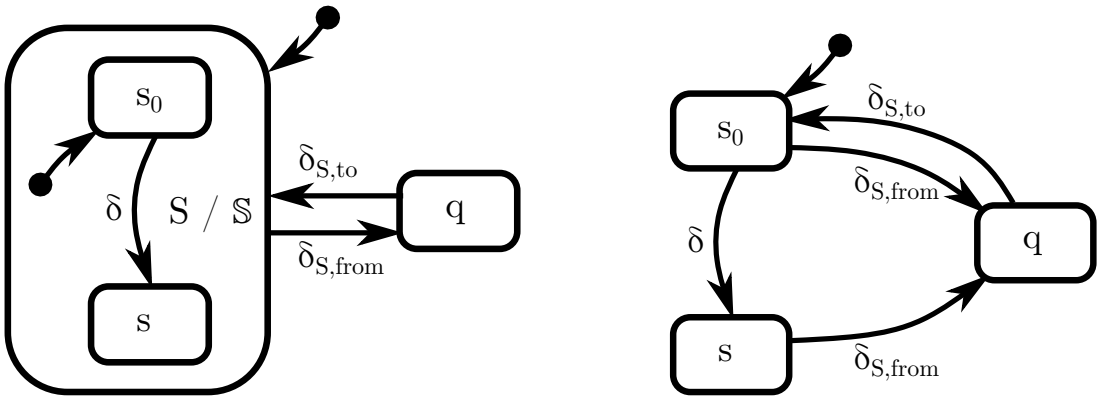
Transformation von orthogonalen und hierarchischen Zustandsautomaten

Im Gegensatz zur Beschreibung der Zustandsautomaten nach Mealy [104] und Moore [105] in Unterabschnitt 3.4.2, werden die hier verwendeten Zustandsautomaten in der Literatur meist als Set von Q, q_0, X, δ, F dargestellt,

- mit der Menge Q an Zuständen: $q_i \in Q$ [106], [107], [130],
- dem Startzustand $q_0 \in Q$ [106], [107], [130],
- der Menge X an Eingabesymbolen: $x_i \in X$ [106], [107], [130],
- der Menge F an Zielzuständen $F \subseteq Q$ [107],
- und der Zustandsübergangsfunktion $\delta : Q \times X \rightarrow F$ bzw. $q = \delta(q, x) \in Q$ [107]

Zur Transformation von orthogonalen und hierarchischen zu einfachen flachen Zustandsautomaten sind einige Regeln notwendig. Dabei wird in der Hierarchie ein Bottom-Up Prinzip angewendet, bei dem zuerst immer die atomaren Subzustände eines Zustandsautomaten aufgelöst werden. Hierbei können in der Hierarchie von Zuständen die einzelnen hierarchischen Zustände auch als einzelne Zustandsautomaten betrachtet werden. In Abbildung 3.9 (a) ist der flache Zustandsautomat S mit den Zuständen $A, B, C \in S$ dargestellt. Sein hierarchisches Äquivalent in Abbildung 3.9 (b) kann als die zwei Zustandsautomaten S und D betrachtet werden, mit den Zuständen $A, C \in D_{Zustandsautomat}$ und $B, D_{Zustand} \in S$ bzw. $D_{Zustandsautomat} \subset S$. Dabei wird D als Zustand $D := D_{Zustand}$ und gleichzeitig als Zustandsautomat $\mathbb{D} := D_{Zustandsautomat}$ mit einer Menge an darunterliegenden Zuständen betrachtet. Eine analoge Sichtweise gilt für die beiden zueinander orthogonalen Zustandsautomaten *SideDoor* (SD) und *EmergencyStop* (ES) in Abbildung 5.5, die als orthogonale Zustände $SD, ES \in \mathbb{P}$,

also in einem übergeordneten orthogonalen Zustandsautomaten $\mathbb{P}$, betrachtet werden können.



(a) Hierarchischer Zustandsautomat (b) Flacher Zustandsautomat

Abbildung 5.7: Formale Beschreibung eines (a) hierarchischen Zustandsautomaten und eines (b) flachen Zustandsautomaten.

Für die Auflösung eines **hierarchischen Zustandsautomaten** $\mathbb{S}$ bzw. Zustand S gilt, entsprechend zu Abbildung 5.7, dass nur die in der Menge $\mathbb{S}$ enthaltenen atomaren Zustände bestehen bleiben. Da dabei der Zustand S verschwindet, werden die Zustandsübergangsfunktionen $\delta_H(S, x_i) = q$, die von diesem hierarchischen Zustand selbst zu einem anderen Zustand $q \in Q$ ausgehen, für alle atomaren Zustände dieses Zustandsautomaten $s \in \mathbb{S}$, auf δ_F repliziert (entsprechend zu Hopcroft et al. [106, S. 61]).

$$q = \delta_H(S, x_i) = \bigcup_{s \in \mathbb{S}} \delta_F(s, x_i)$$

Zustandsübergänge, die von einem Zustand q zu diesem Zustand zeigen $\delta_H(q, x_i) = S$, werden auf den initialen atomaren Zustand $s_0 \in \mathbb{S}$ dieses Zustandsautomaten übertragen.

$$(\delta_H(q, x_i) = S) \Leftrightarrow (\delta_F(q, x_i) = s_0)$$

Alle weiteren Zustandsübergänge bleiben dabei bestehen.

Für die Auflösung eines **orthogonalen Zustandsautomaten** $\mathbb{P}$ bzw. Zustand P gilt, entsprechend zu Abbildung 5.8, dass die orthogonalen Subzustände $P_n \in \mathbb{P}$ als eine Menge von orthogonalen Subzustandsautomaten $\mathbb{P}_n \subset \mathbb{P}$ (mit $n \in N$) betrachtet werden müssen.

$$\mathbb{P}_n = \{p_{n,0}, p_{n,1}, ..., p_{n,|\mathbb{P}_n|-1}\}$$

Damit können durch das kartesische Produkt über alle orthogonalen Subzustandsautomaten, alle möglichen Kombinationen $\mathbb{O}$ der darin enthaltenen atomaren Zustände

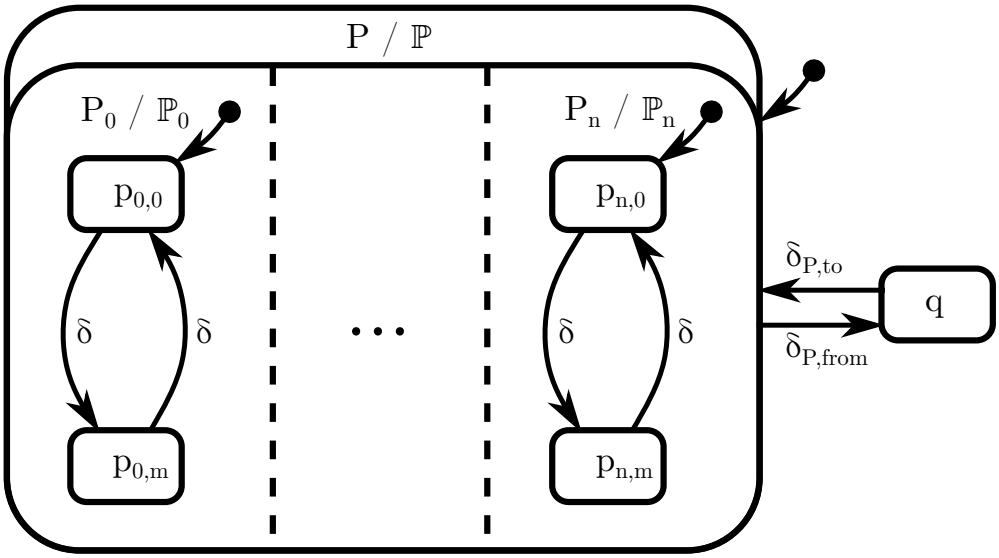


Abbildung 5.8: Formale Beschreibung eines orthogonalen Zustandsautomaten.

$p_{n,m} \in \mathbb{P}_n$ (mit $m \in M$) zwischen diesen orthogonalen Subzustandsautomaten gebildet werden.

$$\begin{aligned}\mathbb{O} = \prod_{n}^{N} \mathbb{P}_n &= \mathbb{P}_0 \times \mathbb{P}_1 \times ... \times \mathbb{P}_n \\ &= \{\{p_{0,0}, p_{1,0}, ...\}, \{p_{0,1}, p_{1,1}, ...\}, ..., \{p_{0,L(0)}, p_{1,L(1)}, ..., p_{n,L(n)}\}\} \\ &= \{\mathbb{O}_0, \mathbb{O}_1, ..., \mathbb{O}_i\}\end{aligned}$$

$$L(n) := \{\ x \in \mathbb{N} \mid 0 \leq x < |\mathbb{P}_n|\}$$

Damit kommt jede mögliche Kombination aus atomaren Zuständen, zwischen den einzelnen Subzustandsautomaten innerhalb der neuen Zustandsautomaten $\mathbb{O}_i \subset \mathbb{O}$ (mit $i \in I$), und ohne Permutationen, genau einmal vor. Dabei ist jeder daraus entstehende Zustandsautomat $\mathbb{O}_i$, wie in Abbildung 5.9 dargestellt, an sich auch ein orthogonaler Zustandsautomat, für den in seinen n orthogonalen Subzuständen $\mathbb{O}_{n,i} \subset \mathbb{O}_i \subset \mathbb{O}$ jeweils nur ein einzelner ($|\mathbb{O}_{n,i}| = 1$) und damit auch gleichzeitig initialer Zustand $p_{n,m} \in \mathbb{O}_{n,i}$ existiert. Für diese kann daher vereinfacht $O_{n,i} \in \mathbb{O}_i$ geschrieben werden. Des Weiteren kann damit auch für den Zustandsautomaten $\mathbb{O}_i$ die vereinfachte Darstellung als Zustand $O_i \in \mathbb{O}$ wie in Abbildung 3.10 (a) oder Abbildung 5.10 verwendet werden. Jeder Zustand O_i bildet damit letztendlich indirekt eine mögliche Kombination aus $I = \prod_n^N |\mathbb{P}_n|$ Einzelzuständen des ursprünglichen orthogonalen Zustandsautomaten $\mathbb{P}$ ab.

$$O_i \Leftrightarrow \mathbb{O}_i = \{p_{0,L(0)}, p_{1,L(1)}, ..., p_{n,L(n)}\}$$

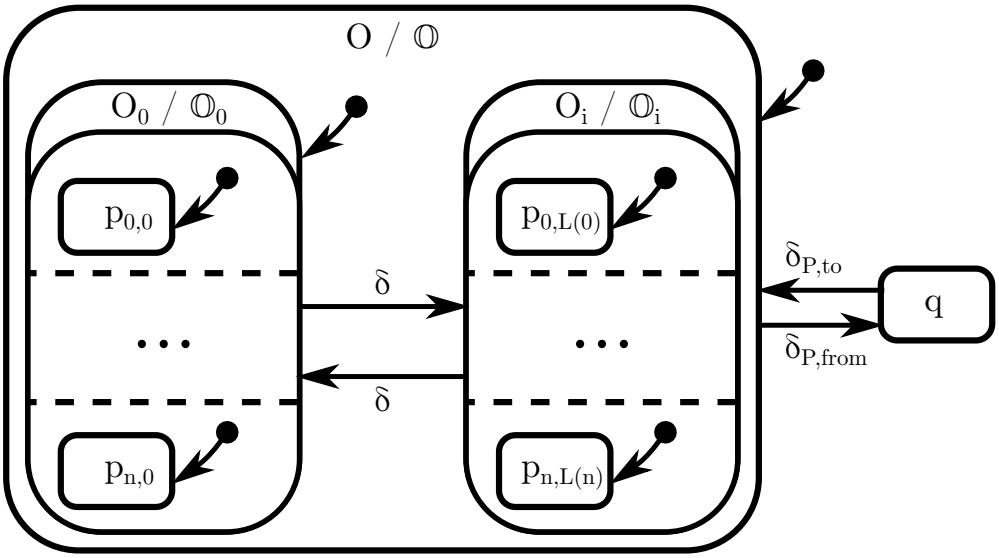


Abbildung 5.9: Formale Beschreibung eines transformierten orthogonalen Zustandsautomaten.

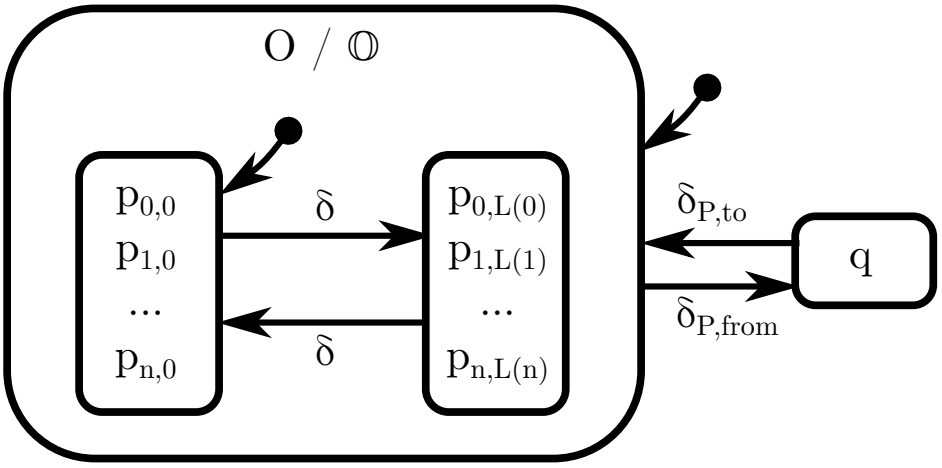


Abbildung 5.10: Vereinfachte Darstellung eines transformierten orthogonalen Zustandsautomaten.

Die Zustandsübergänge $\delta_P : \mathbb{P}_n \times X \rightarrow \mathbb{P}_n$ bzw. $p_{n,k} = \delta_P(p_{n,m}, x)$ zwischen den Zuständen der einzelnen orthogonalen Subzustandsautomaten $p_{n,k}, p_{n,m} \in \mathbb{P}_n$ werden für alle neuen Zustandspaare $O_i, O_j \in \mathbb{O}$ repliziert, wenn diese die ursprünglichen Zustände $p_{n,m/k} \in \mathbb{P}_n \Leftrightarrow x/y \in \mathbb{O}_{j/i}$ in Richtung des Zustandsübergangs enthalten. Gleichzeitig müssen alle anderen Zustände in $\mathbb{O}_{j/i}$ gleich bleiben, da ein Zustandsübergang nur eine

einzelne Zustandsänderung in einem $\mathbb{P}_n$ hervorrufen kann.

$$(p_{n,k} = \delta_P(p_{n,m}, x)) \Leftrightarrow \bigcup_{O_i, O_j \in \mathbb{O}(p_{n,m}, p_{n,k})} (O_i = \delta_O(O_j, x))$$

$$\mathbb{O}(p_{n,m}, p_{n,k}) := \{\ x, y \mid (x_{n,m} \in \mathbb{P}_n \Leftrightarrow x \in \mathbb{O}_j) \ \wedge\ (\{x\} \Leftrightarrow \mathbb{O}_j \setminus \mathbb{O}_i) \wedge (x_{n,k} \in \mathbb{P}_n \Leftrightarrow y \in \mathbb{O}_i) \ \wedge\ (\{y\} \Leftrightarrow \mathbb{O}_i \setminus \mathbb{O}_j)\ \}$$

Aus Abbildung 5.9 und Abbildung 5.10 wird deutlich, dass jeder aus dem orthogonalen Zustandsautomaten $\mathbb{P}$ entstehende Zustandsautomat $\mathbb{O}$ auch eine hierarchische Komponente hat, für welche die vorher genannten Regeln zum Auflösen von hierarchischen Zustandsautomaten $\mathbb{S}$ entspechend auch für $\mathbb{O}$ gelten. Für alle Zustandsübergänge von $\mathbb{O}$ weg, werden diese folglich auf alle O_i übertragen. Für alle Zustandsübergänge zu $\mathbb{O}$ hin, gehen diese stattdessen zum initialen Zustand $O_0 \in \mathbb{O}$, in dem die initialen Zustände aller orthogonalen Subzustände $p_{n,0} \in \mathbb{P}_n$ vorliegen:

$$O_0 := \{\ x \mid x_{n,m} \in \mathbb{O}_i \ \Leftrightarrow\ x_{n,0} \in \mathbb{P}_n\}$$

Der in Abbildung 5.5 dargestellte Zustandsautomat kann durch die Anwendung dieser Regeln in den flachen Zustandsautomaten in Abbildung 5.11 umgewandelt werden. Dabei ergibt sich eine Anzahl von $I = \prod_n^N |\mathbb{P}_n| = |\mathbb{P}_0| \times |\mathbb{P}_1| = 3 \times 2 = 6$ neuen Zuständen $\{O_0, O_1, ..., O_5\} \in \mathbb{O}$ der kombinierten Subzustandsautomaten SD und ES.

$$\begin{aligned}
\{SD.Intermediate, SD.Opened, SD.Closed\} &\in SideDoor \\
&\times \\
\{ES.Triggered, ES.Normal\} &\in EmergencyStop \\
&\Rightarrow \\
\{SD.Intermediate, ES.Triggered\} &\in \mathbb{O}_0 \\
\{SD.Intermediate, ES.Normal\} &\in \mathbb{O}_1 \\
\{SD.Opened, ES.Triggered\} &\in \mathbb{O}_2 \\
\{SD.Opened, ES.Normal\} &\in \mathbb{O}_3 \\
\{SD.Closed, ES.Triggered\} &\in \mathbb{O}_4 \\
\{SD.Closed, ES.Normal\} &\in \mathbb{O}_5
\end{aligned}$$

Die in Tabelle 5.2 indexierten Zustandsübergänge werden dabei auf die neuen Zustände O_i übertragen.

$$\begin{aligned}
\delta_{0,3} : (SD.Closed = \delta_P(&SD.Intermediate, SD.Close[ES.Normal])) \\
\Leftrightarrow ((O_4 = \delta_O(&O_0, SD.Close[ES.Normal])) \\
\cup (O_5 = \delta_O(&O_1, SD.Close[ES.Normal])))
\end{aligned}$$

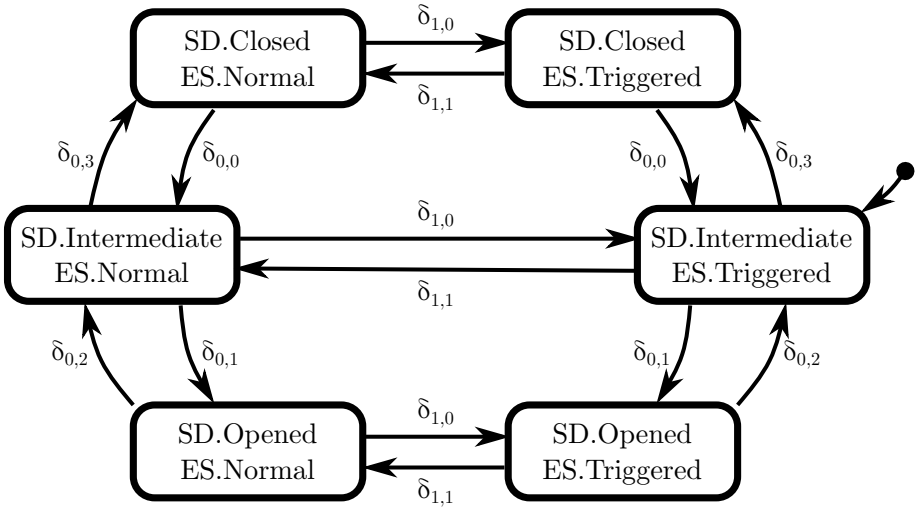


Abbildung 5.11: Umwandlung in einen flachen Zustandsautomaten.

Validierung von Zustandsübergängen in Zustandsautomaten

Betrachtet man Abbildung 5.11 und die darin enthaltenen Zustandsübergänge $\delta_{0,3}$ und deren Bedingungen, wird ersichtlich, dass der Zustandsübergang $O_4 = \delta_O(O_0, SD.Close\text{-}[ES.Normal])$ auf Grund der Bedingung $[ES.Normal]$ nicht möglich ist, da $ES.Normal \notin \mathbb{O}_0$. Die Validierung von Zustandsautomaten ist daher notwendig, um die Gültigkeit von Zustandsübergängen bezüglich deren Bedingungen und Abhängigkeiten von anderen orthogonalen Zuständen zu überprüfen. Dabei gilt für Zustandsübergänge die Darstellung $X : T\ [C]$ mit dem Trigger T und der n-stelligen booleschen Funktion C, welche die für den Zustandsübergang notwendige Bedingung darstellt (entsprechend Guards in UML). Diese wird in SCXML als logische Aussage, mit den Operatoren &&, || und !, und den Variablen $V = \{v_0, v_1, ..., v_n\}$, ausgedrückt. Da die Variablen in den ursprünglichen Zustandsübergängen der orthogonalen Zustandsautomaten auf Zustände in den jeweiligen orthogonalen Subzustandsautomaten verweisen, gilt $V \subset \mathbb{P}$. Für die einzelnen Variablen der Aussage $C(O_j)$ gilt dann, dass diese wahr sind, wenn der Zustand, auf den die jeweilige Variable verweist, auch im Zustandsautomat $\mathbb{O}_j$ indirekt enthalten ist: $\forall x \in V \Rightarrow x \in \mathbb{O}_j$.

Damit kann der Zustandsautomat, durch die Validierung der Zustandsübergänge (unter Vernachlässigung der modellierten internen Signale der Maschine), auf gültige Zustandsübergänge reduziert werden, für die $C(O_j)$ bei der Evaluierung wahr ist und nur noch die Trigger als Eingabesignale notwendig sind. In einem weiteren Schritt werden außerdem die Zustände und deren zugehörige Zustandsübergänge reduziert, die von keinem Zustandsübergang erreicht werden können. Die Reduktion von Abbil-

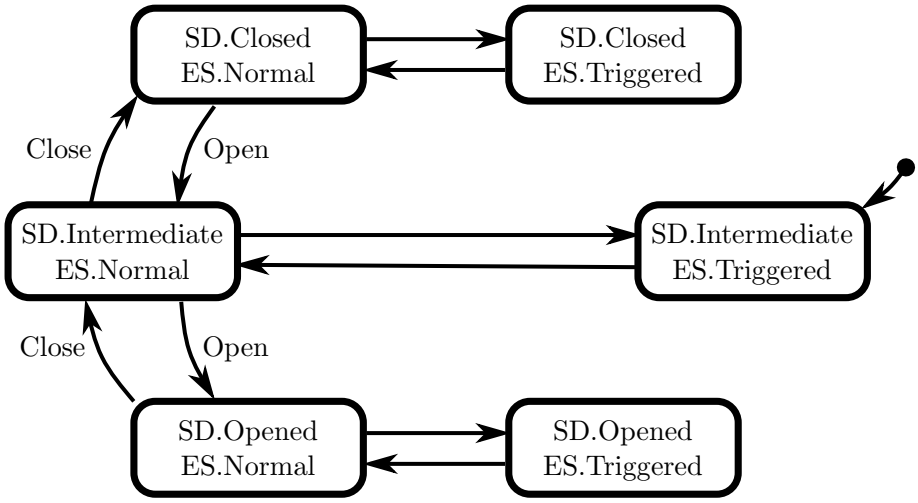


Abbildung 5.12: Validierung der Zustandsübergänge im flachen Zustandsautomaten.

dung 5.11 führt zum Zustandsautomaten in Abbildung 5.12, der die für die Wegfindungsalgorithmen notwendige Struktur und die für die Prozessteuerung notwendigen Informationen enthält. Die Zustandsübergänge ohne Beschriftung können nicht über Trigger ausgelöst werden und sind von internen Signalen abhängig. Damit dürfen diese auch nicht als automatisch auslösende Zustandsübergänge wie in einem ausführbaren Zustandsautomaten interpretiert werden.

Wegfindungsalgorithmen zur dynamischen Prozesssteuerung

Mit der dynamischen Interoperabilität muss es letztendlich möglich sein, in Abhängigkeit von unterschiedlichen Systemzuständen in verschiedenen Systemkomponenten, gemeinsame Ziele im Gesamtsystem zu erreichen. Diese Ziele können u. a. ein einzelner Zustand in einer Systemkomponente, eine Menge an Zuständen in verschiedenen Systemkomponenten oder eine Abfolge von Zuständen sein, die nacheinander erreicht werden müssen. Die jeweiligen aktiven Zustände $\underline{\mathbb{P}_n}$, die in den einzelnen Systemkomponenten jeweils ein Mal vorkommen $\exists! x \forall n(\underline{\mathbb{P}_n} \Rightarrow x \in \mathbb{P}_n)$, dienen dabei als Ausgangsbasis, um den aktuellen Zustand $\underline{\mathbb{O}} = \{\underline{\mathbb{P}_0}, \underline{\mathbb{P}_1}, ..., \underline{\mathbb{P}_n}\}$ im transformierten Zustandsautomaten zu identifizieren. In diesem entsprechen alle enthaltenen Subzustände einem aktiven Systemzustand in einer Komponente $\exists! i \forall x \forall n(x_n \in \mathbb{O}_i \Rightarrow \underline{\mathbb{P}_n})$ Von diesem Zustand ausgehend wird ein Pfad zu einem Zielzustand $Z \in \mathbb{O}$ im transformierten Zustandsautomaten gesucht, der anhand einzelner oder einer Menge seiner enthaltenen Subzustände $\mathbb{Z} = \{Z_0, Z_1, ..., Z_n\} \subseteq \mathbb{O}_i$ identifiziert werden kann.

Der Zustandsautomat entspricht in dieser Form einem Graphen mit unidirektionalen Kanten. Diese werden in der Literatur häufig als $G(V, E)$ mit den Knoten V (*Vertices*) und den Kanten E (*Edges*) bezeichnet [131], [132]. Dabei entspricht V der Menge an resultierenden Zuständen $\mathbb{O}$ und E der Menge an Zustandsübergängen zwischen zwei Zuständen. Ein Pfad in einem solchen Graphen stellt eine endliche Abfolge von Kanten dar, in dem keiner der Knoten (außer möglicherweise der erste und letzte Knoten) mehr als einmal vorkommen [133]. Der Zustandsautomat unterscheidet sich jedoch von den klassischen Darstellungsformen dadurch, dass den Kanten, also den Zustandsübergängen, zusätzlich Trigger hinterlegt sind, und in den Knoten möglicherweise noch implizite Subzustände hinterlegt sind. Bei Graphen, die nur eine Kante zwischen zwei Knoten besitzen können, kann ein Pfad als eine Menge von Kanten $v_0, v_1, ..., v_i$ dargestellt werden [132]. Da der Zustandsautomat aus mehreren unterschiedlichen Zustandsübergängen zwischen zwei Zuständen bestehen kann, muss der Pfad als Menge von Kanten $e_0, e_1, ..., e_i$ dargestellt werden. Außerdem fehlen für viele Wegfindungsalgorithmen die zur Optimierung genutzten Längen der Kanten, daher kann hierfür vorerst nur die Anzahl der passierten Kanten verwendet werden, auch wenn andere Heuristiken zur Optimierung denkbar sind. Klassische Wegfindungsalgorithmen beruhen meist entweder auf einer Breiten- (Breadth First Search (BFS)) oder Tiefensuche (Depth First Search (DFS)). BFS basiert darauf, zuerst immer die Knoten in unmittelbarer Umgebung abzusuchen und sich von dort ebenenweise tiefer zu arbeiten. DFS Algorithmen suchen zuerst von einem einzelnen benachbarten Knoten aus weiter in die Tiefe, bevor sie die restlichen benachbarten Knoten absuchen [131].

Für die Suche nach passenden Pfaden in flachen Zustandsautomaten eignen sich BFS Algorithmen gut, um den kürzesten passenden Zielpfad in unmittelbarer Umgebung zu finden, während DFS Algorithmen eher in Kombination mit geeigneten Heuristiken verwendet werden sollten, um in den einzelnen Knoten die Entfernung oder Wahrscheinlichkeit zu bewerten, mit der bestimmte Kanten zum Ziel führen. Eine Heuristik könnte es z. B. sein gezielt die Kanten zu passieren, deren Zielknoten die gesuchten Subzustände in $\mathbb{O}_i$ enthalten. Mögliche globale Strategien und Bedingungen bei der Wegfindung können außerdem sein:

- Minimierung der Anzahl an Knoten oder Kanten (kürzester Pfad)
- Minimierung der Anzahl der Kanten, die von internen Bedingungen des Systems abhängen
- nur Knoten passieren, die bestimmte Subzustände in $\mathbb{O}_i$ enthalten, um gewisse Systemzustände in den Komponenten zu erhalten
- Zwischenziele in einer bestimmten Reihenfolge zu passieren

- zusätzliche Heuristiken zur Optimierung durch Auswertung am realen System bestimmen

5.2.5 Konzeptuelle Interoperabilität

Wie bereits im vorherigen Abschnitt erwähnt, beschreibt die konzeptuelle Interoperabilität die abstrakte Anwendung und prinzipielle Kooperation einer Maschine mit anderen Komponenten des Gesamtsystems. Alle Komponenten des Systems enthalten dabei Teilwissen, das auf Grund von gemeinsamen Eigenschaften und Prinzipien zusammengefasst werden muss. Dieses Teilwissen und dessen Zusammenhang mit anderen Komponenten muss in den einzelnen Komponenten mit seiner Semantik und Pragmatik beschrieben und auf konzeptueller Ebene verknüpft werden. Die konzeptuelle Ebene enthält das dazu notwendige Wissen:

1. Die **Struktur** des Gesamtsystems und die Verknüpfung der Komponenten auf Basis von bekannten Standardanwendungen und dem Teilwissen der Komponenten: Die Struktur bezieht sich dabei einerseits auf die tatsächliche physische Zusammensetzung des Systems aus Maschinen und Komponenten, andererseits auf die daraus resultierende Verknüpfung der Pragmatik der einzelnen Komponenten über unterschiedliche Maschinen hinweg, welche durch die Semantik unterstützt wird.
2. Die **Composed Skills** des Gesamtsystems auf Basis der Fähigkeiten der Teilsysteme und der Komposition dieser im Gesamtsystem: Die Komposition basiert auf der Struktur und Verknüpfung der Anlagenkomponenten und erlaubt die Kombination der einzelnen Fähigkeiten dieser Komponenten, um neue und komplexere Fähigkeiten im Gesamtsystem bereitzustellen. Die konzeptuelle Interoperabilität enthält die dazu notwendigen abstrakten Beschreibungen von Fähigkeiten und Skills der einzelnen Maschinen und nacheinander zu erreichenden Zustände in diesen.
3. Die **Ziele** des Gesamtsystems und **Subziele** in den einzelnen Komponenten: Die gesamte Anlage hat den Zweck, bestimmte Aufgaben auf Grund ihrer Fähigkeiten (Composed Skills) zu erfüllen. Die Gesamtziele stellen die dabei zu erreichenden Zielzustände in der Anlage dar. Subziele können dabei notwendige Teilzustände in den einzelnen Maschinen und Komponenten sein.

Die konzeptuelle Interoperabilität greift dabei auf Teilwissen aus der Pragmatik der einzelnen Maschinen zurück. Genauer gesagt können Zusammenhänge zwischen den

einzelnen Komponenten auf pragmatischer Ebene beschrieben werden, die eigentlich der konzeptuellen Ebene zuzuordnen sind. Dazu zählt die Möglichkeit, Abhängigkeiten von externen Zuständen als Bedingungen für Zustandsübergänge zu modellieren, die beim Betrieb als Einzelmaschine nicht berücksichtigt werden müssen, aber bei der Verkettung mit anderen Maschinen eine Rolle spielen. Eine solche Bedingung könnte bei der Beladetür einer Werkzeugmaschine hinzugefügt werden, um die Abhängigkeit von der Position eines Handhabungssystems beim Schließen der Beladetür zu hinterlegen. Die zusätzliche Abhängigkeit *WorkpieceHandling.Outside* beim Zustandsübergang *SideDoor.Opened* zu *SideDoor.Closed* ist der Herstellerfirma der Maschine schon bei der Entwicklung und Programmierung der Maschine bekannt und wird teilweise auch als notwendiges Signal bei herkömmlichen Handhabungsschnittstellen von Maschinen verlangt. Da das Signal aber von einem externen und austauschbaren Gerät bereitgestellt werden muss, ist dieses Wissen Teil der konzeptuellen Ebene. Eine Zellensteuerung kann sich dieses Wissen zu Nutze machen und die Inbetriebnahme erleichtern, indem notwendige Signale zwischen den unterschiedlichen Maschinen und Komponenten innerhalb der Zelle (teil-)automatisch verknüpft werden. Dazu ist jedoch einerseits eine eindeutige Zuordnung der Zustände notwendig, die auf einer einheitlichen Semantik zwischen den Maschinen basieren muss, und andererseits wird die Information zur strukturellen Verknüpfung der Komponenten benötigt, um notwendige Signale den korrekten Komponenten zuordnen zu können, wenn z. B. im vorher genannten Fall mehrere Handhabungssysteme in der Anlage vorhanden sind, aber nur eines davon die Maschine überhaupt beladen kann.

Konzeptuelle Zusammenhänge können aber auch als komplette Zustandsautomaten modelliert werden, die das Verhalten eines externen Systems in Form von dessen Zuständen und Zustandsübergängen darstellen, um die Beziehung zu internen Komponenten für bestimmte Aktionen zu beschreiben. Das ist besonders bei einem Einsatz eines Roboters für die Werkstückbeladung geeignet, der flexibel gestaltbar ist und frühestens bei einer virtuellen Inbetriebnahme der Fertigungszelle mit den anwendungsspezifischen Positionen und Pfaden programmiert wird. Die Werkzeugmaschine bringt dabei das Teilwissen zu Positionen, Zustandsänderungen und den dazu notwendigen Bedingungen der Werkzeugmaschine selbst mit.

Letztendlich muss auf dieser Ebene aber auch generelles Domänenwissen vorhanden sein, mit dem grundsätzliche Fähigkeiten, Regeln und Zusammenhänge des Gesamtsystems abgebildet werden, aus denen die Komposition der Maschinen im Fertigungssystem ermöglicht wird. Die Komposition von Werkzeugmaschine und Handhabungssystem ermöglicht z. B. das Be- und Entladen von Werkstücken, wobei die Werkzeugmaschine alleine nur die Fähigkeit zur Bearbeitung von Werkstücken mitbringt. Das Ziel zur vollautomatisierten Bearbeitung eines bestimmten Werkstücks kann erreicht werden, wenn die Subziele zur Beladung, Bearbeitung und Entladung

des Werkstücks nacheinander erreicht werden, bzw. die dazugehörigen Skills der entsprechenden Maschinen korrekt ausgeführt werden. Die Kombination dieses Wissens mit dem Teilwissen der Einzelmaschinen ermöglicht die Zusammenführung dieser abstrakten Ziele mit den speziellen Eigenschaften, Fähigkeiten und Bedingungen, also der Pragmatik, einzelner Maschinen, ohne diese zu sehr auf Standards einzuschränken.

5.3 Der Weg zur agilen Automatisierung

In dieser Arbeit liegt der Fokus auf der pragmatischen und der dynamischen Ebene, da die restlichen Ebenen bereits umfassend erforscht sind. Diese beiden Ebenen ermöglichen mit der Modellierung des Verhaltens von Maschinen auch die dynamische Konfiguration und Steuerung dieser innerhalb eines automatisierten Fertigungssystems. Damit erlauben sie eine Vereinfachung der Programmierung und Inbetriebnahme von derartigen Fertigungssystemen, bis hin zur automatisierten Inbetriebnahme von nichtstandardisierten Maschinen. Die dazu in Abbildung 5.4 beschriebenen Bausteine zeigen aktuelle Technologien, die sich für diesen Einsatz eignen, aber grundsätzlich mit anderen Technologien, die vergleichbare Funktionalitäten bieten, austauschbar sind. Die folgenden Kapitel beschäftigen sich mit der pragmatischen Modellierung des Verhaltens und der Abhängigkeiten einer Werkzeugmaschine, der zusätzlichen Beschreibung eines externen Handhabungsroboters (Kapitel 6), und der Evaluierung dieser Modellierung hinsichtlich der dynamischen Interoperabilität, zur automatisierten Identifikation von Prozessen (Kapitel 7).

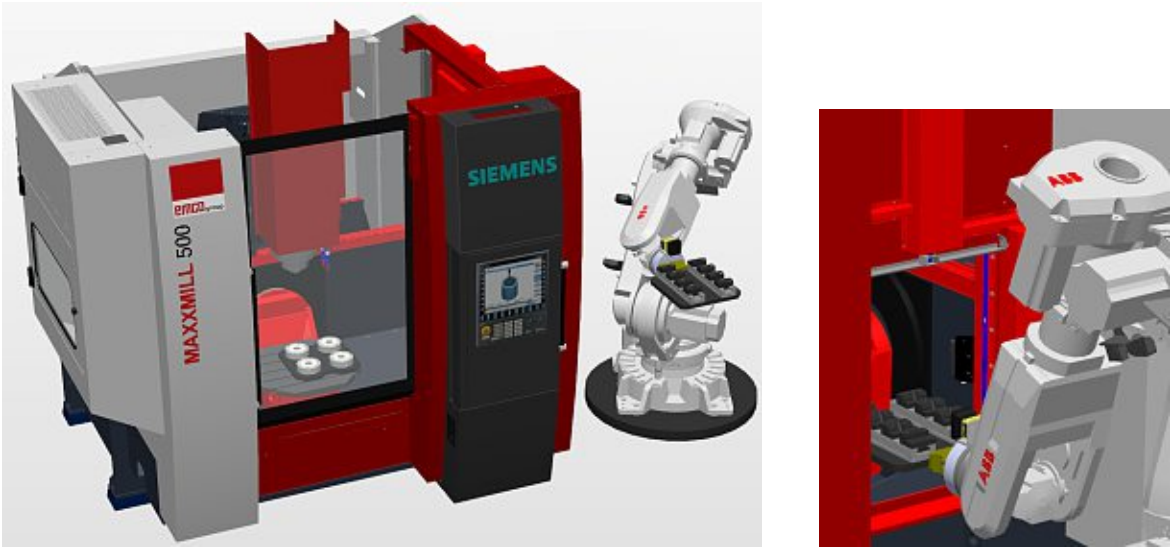
Kapitel 6

Implementierung und Proof of Concept

Um die im vorhergehenden Kapitel 5 beschriebenen Methoden zur Modellierung der Pragmatik zu testen, und daraufhin im folgenden Kapitel 7 hinsichtlich der Regeln zur dynamischen Interoperabilität zu evaluieren, wird in diesem Kapitel die Pragmatik einer Werkzeugmaschine für den Einsatz in einer roboterisierten Fertigungszelle modelliert.

6.1 Fertigungszelle

Abbildung 6.1 zeigt die Fertigungszelle aus Sicht der Werkzeugmaschine. Die Zelle enthält eine Fräsmaschine zur 5-Achs-Bearbeitung von Werkstücken mit Werkzeugmagazin und -wechsler und der Möglichkeit zur Be- und Entladung von Werkstückpaletten mit Handhabungssystemen über die seitliche Beladetür. Abbildung 6.2 zeigt dazu die Komponenten der Fertigungszelle, die für die Automatisierung dieser Funktionen aus Sicht der Werkzeugmaschine notwendig sind. Für die **automatisierte Bearbeitung** selbst ist die numerische Steuerung *NumericalControl* zuständig, die im automatischen *OperationMode*, bei aktivierten Antrieben *AuxiliaryDrives* und inaktivem Not-Halt *EmergencyStop*, Bearbeitungsprogramme ausführen kann. Für die **Handhabung von Werkstücken** existiert ein Nullpunktspannsystem *ClampingSystem*, mit dem Werkstückpaletten auf dem Bearbeitungstisch *MachiningTable* punktgenau gespannt werden können. Über die Beladetür *SideDoor* werden diese Paletten von



(a) Fertigungszelle (b) Beladung

Abbildung 6.1: (a) Fertigungszelle (b) Beladetür.

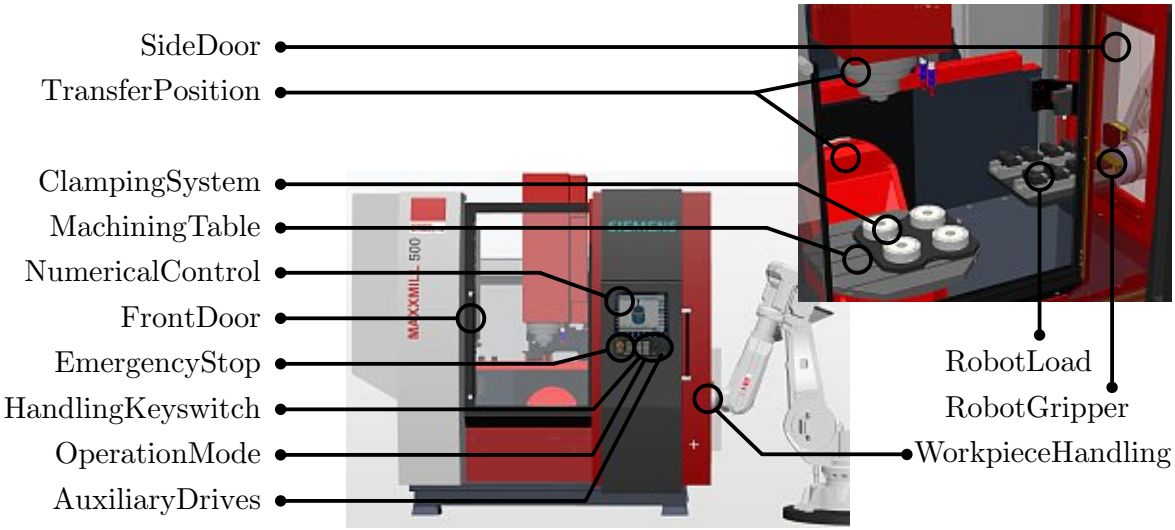


Abbildung 6.2: Maschinen und Subkomponenten.

einem Handhabungssystem be- und entladen. Dafür müssen jedoch der Handschalter $HandlingKeyswitch$ für die Handhabungsschnittstelle manuell aktiviert worden sein und sich die Achsen der Maschine in einer fest vorgeschriebenen Position $TransferPosition$ befinden, in der eine Kollision der Handhabungseinrichtung mit den Maschinenkomponenten ausgeschlossen werden kann. Aus Sicht der Werkzeugmaschine sind drei externe Komponenten für die Handhabung relevant. $WorkpieceHandling$ beschreibt

die aktuelle Position des Roboters außerhalb und innerhalb der Maschine. Dazu kommen mit *RobotLoad* und *RobotGripper* der Belade- und Spannzustand des Greifers.

6.2 Werkzeugmaschine

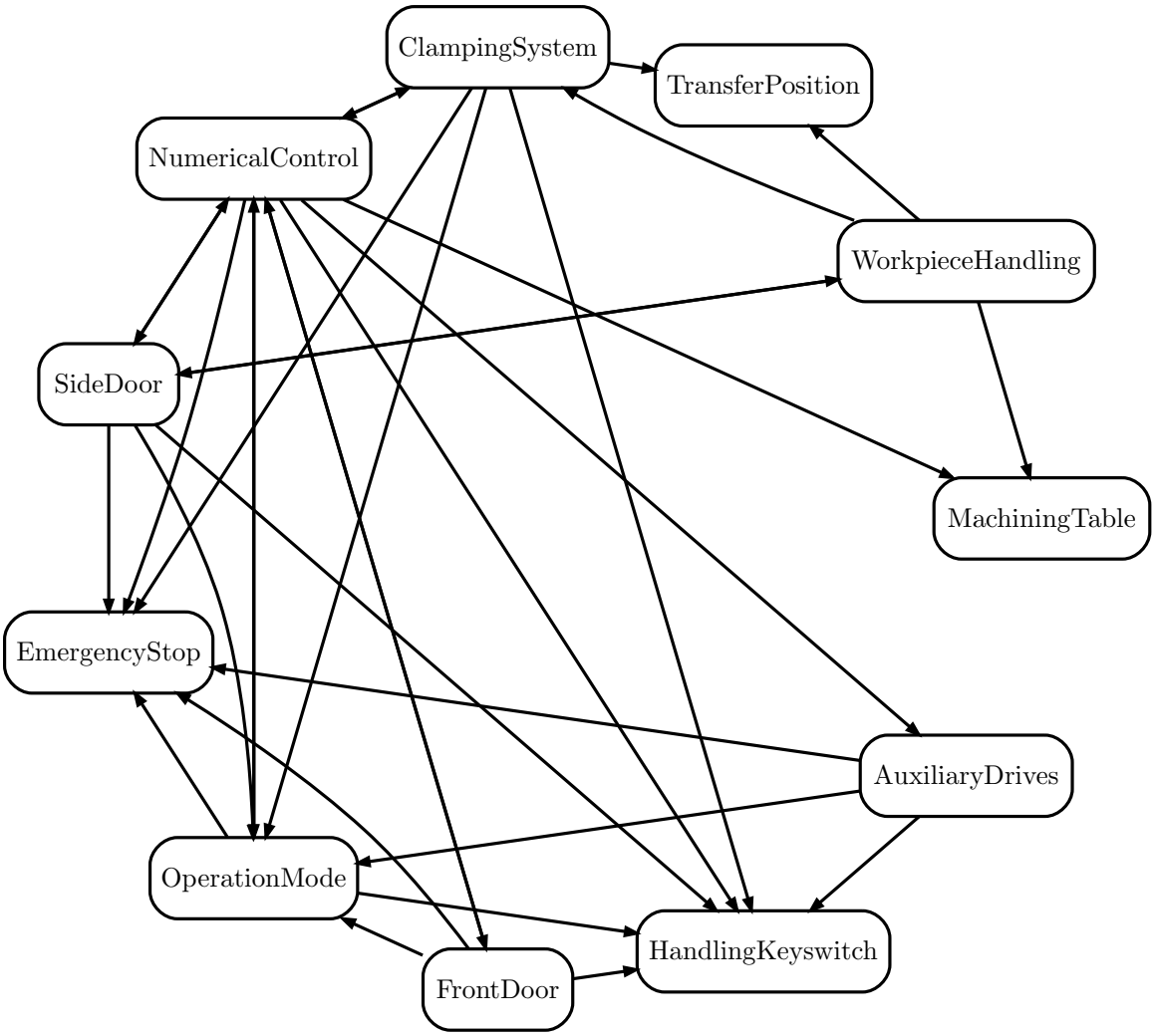


Abbildung 6.3: Abhängigkeiten zwischen den Komponenten der Werkzeugmaschine.

Die im Folgenden modellierten Komponenten der Werkzeugmaschine (ClampingSystem (CS), SideDoor (SD), NumericalControl (NC), FrontDoor (FD), AuxiliaryDrives (AD),

EmergencyStop (ES), HandlingKeyswitch (HK), OperationMode (OM), TransferPosition (TP) und MachiningTable (MT)) stehen entsprechend Abbildung 6.3 in einer starken Abhängigkeit zueinander. Außerdem gibt es direkte Abhängigkeiten zu den externen Komponenten RobotLoad (RL), WorkpieceHandling (WH), die in den kommenden Unterabschnitten beschrieben sind.

6.2.1 ClampingSystem (CS)

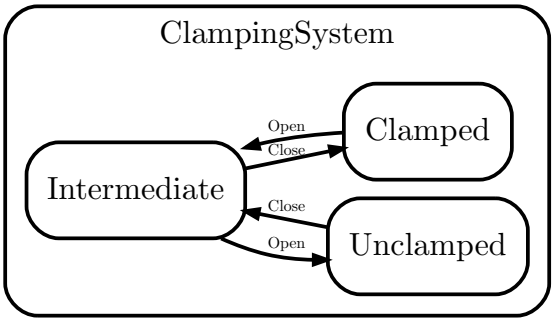


Abbildung 6.4: Zustandsautomat ClampingSystem (CS) [1].

Der Zustandsautomat der Komponente CS in Abbildung 6.4, mit den Bedingungen in Tabelle 6.1, kann die Zustände *Intermediate*, *Unclamped* und *Clamped* annehmen. Die dazugehörigen Fähigkeiten sind *Close* und *Open*. Das Nullpunktspannsystem kann entweder aktiv gespannt *Clamped* oder aktiv entspannt *Unclamped* werden. Liegt kein Signal an, so befindet es sich im Zustand *Intermediate*, in dem jedoch die Federn im Spannsystem die Werkzeugpalette trotzdem fixieren, aber die Spannkraft zur Bearbeitung von Werkstücken möglicherweise nicht ausreicht. Neben einigen internen Abhängigkeiten von anderen Komponenten, ist bei Nutzung der Roboterschnittstelle zur Be- und Entladung notwendig, dass sich der Handhabungsroboter an der dazu notwendigen Position $WorkpieceHandling.Table$ befindet.

Tabelle 6.1: Abhängigkeiten von ClampingSystem (CS).

Von	Nach	Skill	Bedingungen
Intermediate	Unclamped	Open	!NumericalControl.Running
Clamped	Intermediate	Open	EmergencyStop.Normal

Von	Nach	Skill	Bedingungen
			HandlingKeyswitch.On
			OperationMode.Auto
			TransferPosition.Reached
Intermediate	Clamped	Close	!NumericalControl.Running
Unclamped	Intermediate	Close	EmergencyStop.Normal
			HandlingKeyswitch.On
			OperationMode.Auto
			TransferPosition.Reached
			MachiningTable.Occupied

6.2.2 SideDoor (SD)

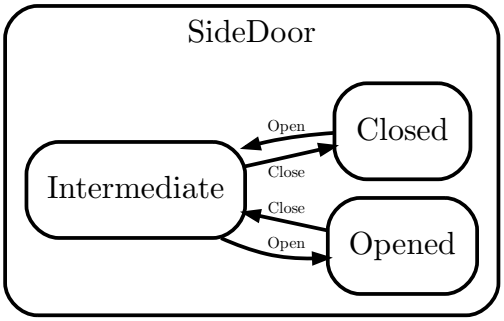


Abbildung 6.5: Zustandsautomat SideDoor (SD) [1].

Der Zustandsautomat der Komponente SD in Abbildung 6.5, mit den Bedingungen in Tabelle 6.2, kann die Zustände *Intermediate*, *Opened* und *Closed* annehmen. Die dazugehörigen Fähigkeiten sind *Close* und *Open*. Damit kann die Beladetür aktiv geöffnet und geschlossen werden. Die Zustände werden durch die Endschalter an der Tür bestimmt und resultieren im Zustand *Intermediate*, wenn keiner der beiden Endschalter berührt wird. Dieser Zustand sollte aber (außer im Fehlerzustand) immer nur kurzfristig während dem Öffnen und Schließen der Tür anliegen. Die Tür besitzt außerdem die Abhängigkeit $WorkpieceHandling.Outside$ zum Schließen, damit der

Handhabungsroboter dabei nicht den Türbereich versperrt.

Tabelle 6.2: Abhängigkeiten von SideDoor (SD).

Von	Nach	Skill	Bedingungen
Intermediate	Opened	Open	!NumericalControl.Running
Closed	Intermediate	Open	EmergencyStop.Normal
			HandlingKeyswitch.On
			OperationMode.Auto
Intermediate	Closed	Close	!NumericalControl.Running
Opened	Intermediate	Close	EmergencyStop.Normal
			HandlingKeyswitch.On
			OperationMode.Auto
			WorkpieceHandling.Outside

6.2.3 FrontDoor (FD)

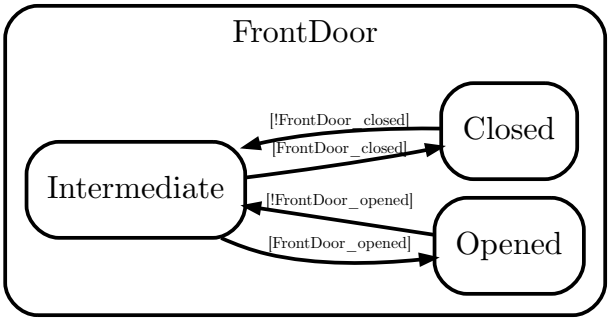


Abbildung 6.6: Zustandsautomat FrontDoor (FD).

Der Zustandsautomat der Komponente FD in Abbildung 6.6, mit den Bedingungen in Tabelle 6.3, kann die Zustände *Intermediate*, *Opened* und *Closed* annehmen. Er entspricht dem Zustandsautomaten SideDoor (SD), jedoch können Zustandänderungen bei dieser Maschinentür nur manuell hervorgerufen werden.

Tabelle 6.3: Abhängigkeiten von FrontDoor (FD).

Von	Nach	Skill	Bedingungen
Intermediate	Opened		!NumericalControl.Running
Intermediate	Closed		EmergencyStop.Normal
Opened	Intermediate		HandlingKeyswitch.On
Closed	Intermediate		OperationMode.Auto

6.2.4 NumericalControl (NC)

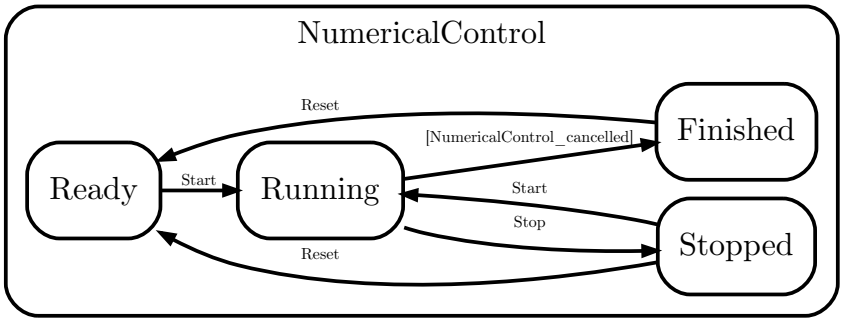


Abbildung 6.7: Zustandsautomat NumericalControl (NC) [1].

Der Zustandsautomat der Komponente NC in Abbildung 6.7, mit den Bedingungen in Tabelle 6.4, kann die Zustände *Ready*, *Stopped*, *Running* und *Finished* annehmen. Die dazugehörigen Fähigkeiten sind *Reset*, *Start* und *Stop*. Die numerische Steuerung dient zur Ausführung von Bearbeitungsprogrammen, mit denen die einzelnen Achsen der Maschine einzeln oder in Kombination angesteuert werden, um ein Werkstück mit einem in der Spindel eingespannten Werkzeug zu bearbeiten. Um ein solches Programm zu starten und damit in den Zustand *Running* zu kommen, müssen mehrere Bedingungen erfüllt sein: alle Maschinentüren (*FrontDoor*, *SideDoor*) und Spannmittel (*ClampingSystem*) sind geschlossen, und die Maschine ist zur Bearbeitung bereit (*AuxiliaryDrives.On*, *OperationMode.Auto*, *EmergencyStop.Normal*, *HandlingKeyswitch.On*, *MachiningTable.Occupied*). Während der Bearbeitung kann ein solches Programm angehalten (*Stopped*) und wieder gestartet (*Running*) oder zurückgesetzt (*Ready*) werden. Ist ein Programm komplett abgelaufen, wechselt der Zu-

standsautomat in den Zustand *Finished*, der nicht aktiv aufgerufen werden kann, sondern erst nach der Fertigstellung der Bearbeitung (und damit auch nur vom Zustand *Running* aus) automatisch eintritt.

Tabelle 6.4: Abhängigkeiten von NumericalControl (NC).

Von	**Nach**	**Skill**	**Bedingungen**
Stopped	Ready	Reset	EmergencyStop.Normal
Running	Stopped	Stop	HandlingKeyswitch.On
Running	Finished		OperationMode.Auto
Finished	Ready	Reset	
Ready	Running	Start	EmergencyStop.Normal
Stopped	Running	Start	HandlingKeyswitch.On
			OperationMode.Auto
			ClampingSystem.Clamped
			SideDoor.Closed
			FrontDoor.Closed
			AuxiliaryDrives.On
			MachiningTable.Occupied

6.2.5 AuxiliaryDrives (AD)

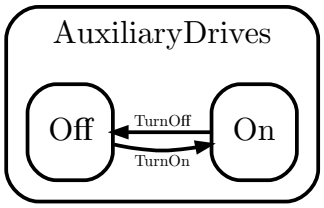


Abbildung 6.8: Zustandsautomat AuxiliaryDrives (AD) [1].

Der Zustandsautomat der Komponente AD in Abbildung 6.8, mit den Bedingungen in Tabelle 6.5, kann die Zustände Off und On annehmen. Die dazugehörigen Fähigkeiten sind $TurnOff$ und $TurnOn$. Dieser Zustandsautomat beschreibt den Zustand der Antriebe der Werkzeugmaschine. Diese müssen bei der Bearbeitung aktiviert sein.

Tabelle 6.5: Abhängigkeiten von AuxiliaryDrives (AD).

Von	Nach	Skill	Bedingungen
Off	On	TurnOn	EmergencyStop.Normal
On	Off	TurnOff	HandlingKeyswitch.On
			OperationMode.Auto

6.2.6 HandlingKeyswitch (HK)

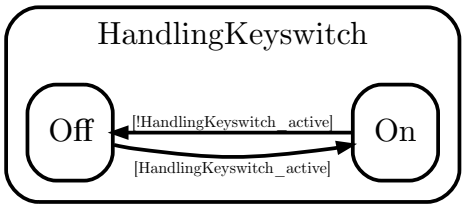


Abbildung 6.9: Zustandsautomat HandlingKeyswitch (HK).

Der Zustandsautomat der Komponente HK in Abbildung 6.9 kann die Zustände Off und On annehmen. Der Schlüsselschalter muss manuell vom Bediener in den Zustand On gebracht werden, damit die externe Steuerung der Maschine über die Schnittstelle freigegeben wird.

6.2.7 EmergencyStop (ES)

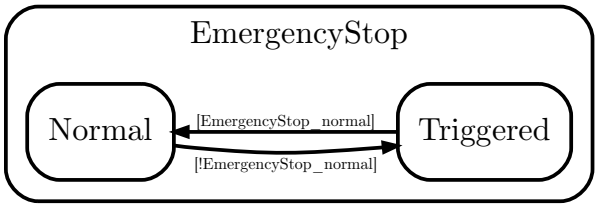


Abbildung 6.10: Zustandsautomat EmergencyStop (ES).

Der Zustandsautomat der Komponente ES in Abbildung 6.10 kann die Zustände *Normal* und *Triggered* annehmen. Der Zustand *Normal* für den Not-Halt ist als Bedingung für fast alle Zustandsübergänge in anderen Komponenten notwendig, um eine sichere Bedienung der Maschine zu ermöglichen. Der Not-Halt kann meist nur manuell in diesen Zustand gebracht werden und ist daher auch so modelliert.

6.2.8 OperationMode (OM)

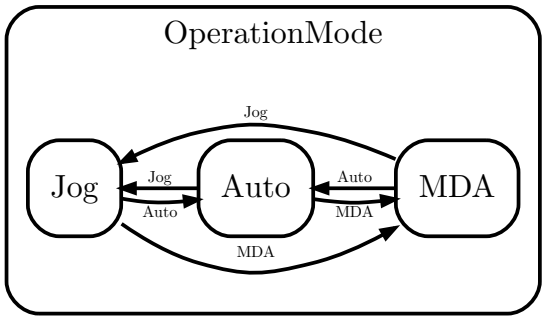


Abbildung 6.11: Zustandsautomat OperationMode (OM).

Der Zustandsautomat der Komponente OM in Abbildung 6.11, mit den Bedingungen in Tabelle 6.6, kann die Zustände *Jog*, *Auto* und *MDA* annehmen. Die dazugehörigen Fähigkeiten sind *Auto*, *Jog* und *MDA*. Die Werkzeugmaschine erlaubt die Bearbeitung von Werkstücken in verschiedenen Modi, von denen *Jog* das manuelle Verfahren

von Achsen ermöglicht und *MDA* die Ausführung von CNC Einzelbefehlen. Der Zustand *Auto* ist letztendlich für die Abarbeitung von CNC Programmen notwendig.

Tabelle 6.6: Abhängigkeiten von OperationMode (OM).

Von	Nach	Skill	Bedingungen
Jog	Auto	Auto	!NumericalControl.Running
Jog	MDA	MDA	EmergencyStop.Normal
Auto	Jog	Jog	HandlingKeyswitch.On
Auto	MDA	MDA	
MDA	Auto	Auto	
MDA	Jog	Jog	

6.2.9 TransferPosition (TP)

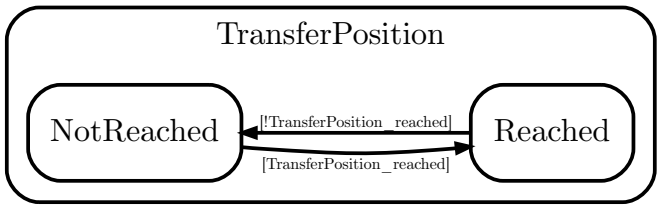


Abbildung 6.12: Zustandsautomat TransferPosition (TP).

Der Zustandsautomat der Komponente TP in Abbildung 6.12, mit den Bedingungen in Tabelle 6.7, kann die Zustände *NotReached* und *Reached* annehmen. Diese ist weniger eine Komponente der Maschine als ein notwendiger Zustand für die Handhabung von Werkstücken. Der Zustand wird erreicht, wenn sich die einzelnen Maschinenachsen auf vordefinierten Positionen befinden, um eine Kollision bei der Be- und Entladung zu vermeiden. Dieser Zustand wird oft durch die Ausführung vorprogrammierter CNC Programme hergestellt und wird daher meist an das Ende von Bearbeitungsprogrammen angehängt, um die Maschine automatisch in den erforderlichen Zustand zu versetzen. Dieses Signal ist hier nur intern modelliert und kann nicht von außen angefordert werden, da die Ausführung von CNC Programmen in unbekannten Aufspannsituationen zu Kollisionen führen kann. Trotzdem ist das Signal notwendig, um

eine sichere Be- und Entladung zu gewährleisten. Damit ist es für diese Anwendungen notwendig diesen Zustand herzustellen, der aber als internes Signal nur als Bedingung überprüft werden kann. Trotzdem wurden in Tabelle 6.7 Bedingungen definiert, die für den Zustandswechsel zwischen *Reached* und *NotReached* notwendig sind. Diese können den Steuerungsalgorithmus durch die zusätzliche Information unterstützen, unter welchen Bedingungen ein gewisser Zustand hergestellt werden kann, auch wenn dieser Zustandsübergang keine direkte Folge der erfüllten Bedingungen ist.

Tabelle 6.7: Abhängigkeiten von TransferPosition (TP).

Von	**Nach**	**Skill**	**Bedingungen**
NotReached	Reached		SideDoor.Closed
Reached	NotReached		FrontDoor.Closed
			EmergencyStop.Normal
			HandlingKeyswitch.On

6.2.10 MachiningTable (MT)

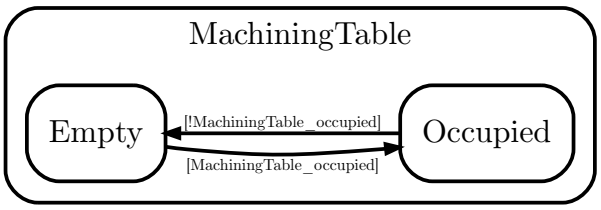


Abbildung 6.13: Zustandsautomat MachiningTable (MT).

Der Zustandsautomat der Komponente MT in Abbildung 6.13, mit den Bedingungen in Tabelle 6.8, kann die Zustände *Empty* und *Occupied* annehmen. Dieser Zustandsautomat beschreibt den Beladezustand der Werkzeugmaschine mit einer Werkstückpalette auf dem Bearbeitungstisch. Dieser Zustandsautomat ist für den Bewegungsablauf des Handhabungsroboters in WorkpieceHandling (WH) notwendig. Der Zustand kann zwar nicht über einen Skill verändert werden, sondern ist abhängig von einem induktiven Näherungssensor auf dem Maschinentisch, aber die für die Änderung notwendigen Bedingungen können (ähnlich zu Unterabschnitt 6.2.9) trotzdem modelliert werden, um

die Steuerungsalgorithmus bei der Identifikation der Prozessschritte zu unterstützen.

Tabelle 6.8: Abhängigkeiten von MachiningTable (MT).

Von	Nach	Skill	Bedingungen
Empty	Occupied		!NumericalControl.Running
Occupied	Empty		EmergencyStop.Normal
			HandlingKeyswitch.On
			OperationMode.Auto
			TransferPosition.Reached

6.3 Handhabungsroboter

Die Modellierung des Handhabungsroboters enthält nur die für die Automatisierung, aus Sicht der Werkzeugmaschine notwendigen Komponenten RobotGripper (RG), RobotLoad (RL) und WorkpieceHandling (WH). Diese bilden den Belade- und Spannzustand des Robotergreifers und dessen Position ab, um Abhängigkeiten mit den Komponenten der Werkzeugmaschine zu beschreiben. Diese sind zur Übersicht in Abbildung 6.14 dargestellt. Daraus werden die Zusammenhänge zu den bei der Beladung notwendigen Komponenten *ClampingSystem*, *MachiningTable* und $TransferPosition$ sichtbar, die letztendlich bei der Steuerung des Handhabungsroboters berücksichtigt werden müssen.

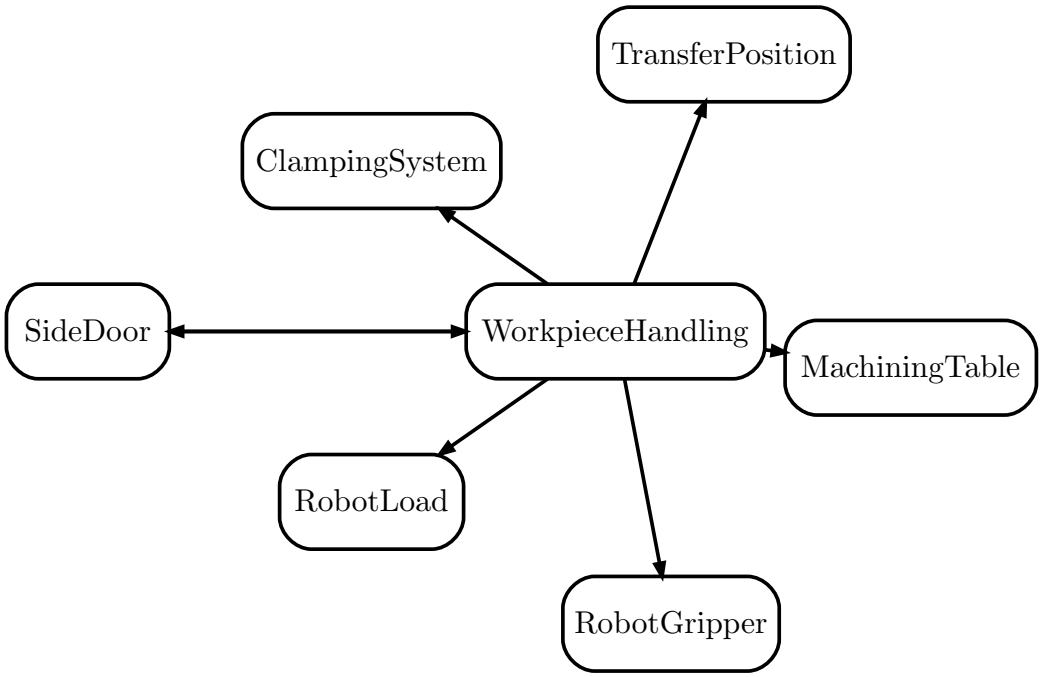


Abbildung 6.14: Abhängigkeiten zwischen den Komponenten des Handhabungsroboters.

6.3.1 RobotGripper (RG)

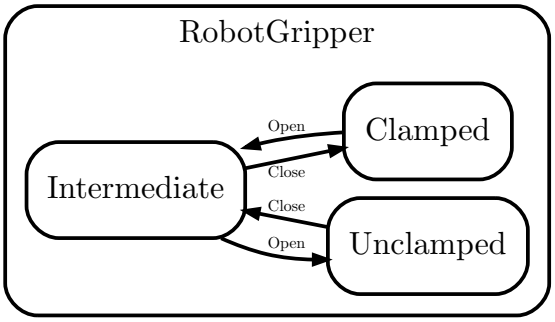


Abbildung 6.15: Zustandsautomat RobotGripper (RG) [1].

Der Zustandsautomat der Komponente RG in Abbildung 6.15 kann die Zustände *Intermediate*, *Unclamped* und *Clamped* annehmen. Dieser entspricht inhaltlich dem

ClampingSystem (CS) Zustandsautomaten der Werkzeugmaschine. Der Greifer kann ebenfalls aktiv ge-und entspannt werden, und ist im Ruhezustand *Intermediate* ebenfalls noch über Federn gespannt. Der Robotergreifer kann damit geeignete Werkstückpaletten aufnehmen und fixieren, die aber bei höheren Anforderungen (z. B. hinsichtlich Gewicht der Palette und Geschwindigkeit des Roboters bei der Bewegung) aktiv gespannt werden müssen.

6.3.2 RobotLoad (RL)

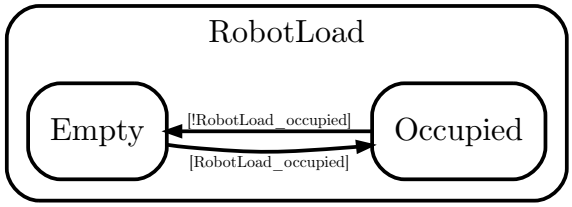


Abbildung 6.16: Zustandsautomat RobotLoad (RL).

Dieser entspricht dem MachiningTable (MT) Zustandsautomaten der Werkzeugmaschine, und stellt den induktiven Näherungssensor im Robotergreifer dar, der anzeigt, ob sich eine Werkstückpalette im Greifer befindet. Diese Information ist hauptsächlich für die möglichen Fahrbahnen des Roboters in WorkpieceHandling (WH) in Kombination mit der Komponente MachiningTable (MT) notwendig. Der Zustandsautomat der Komponente RL in Abbildung 6.16 kann die Zustände *Empty* und *Occupied* annehmen.

6.3.3 WorkpieceHandling (WH)

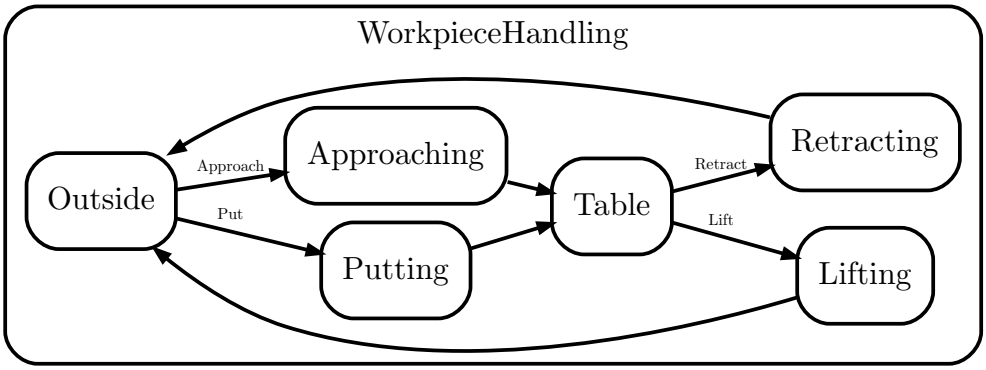


Abbildung 6.17: Zustandsautomat WorkpieceHandling (WH) [1].

Der Zustandsautomat der Komponente WH in Abbildung 6.17, mit den Bedingungen in Tabelle 6.9, kann die Zustände *Outside*, *Putting*, *Lifting*, *Approaching*, *Retracting* und *Table* annehmen. Die dazugehörigen Fähigkeiten sind *Approach*, *Lift*, *Put* und *Retract*. Der Zustandsautomat beschreibt die möglichen Positionen und Bewegungspfade des Handhabungsroboters in Kombination mit den Komponenten der Werkzeugmaschine. Dabei wird zwischen den beiden Positionen des Roboters außerhalb der Maschine (*Outside*) und innerhalb der Maschine mit dem Greifer am Maschinentisch (*Table*) unterschieden. Dazwischen existieren vier mögliche Pfade, um zwischen diesen Positionen zu verfahren, die jedoch abhängig von der geplanten Aufgabe, also der Be- oder Entladung der Werkzeugmaschine, sind. Zur Beladung der Maschine befindet sich der Roboter initial im Zustand *Outside* und sollte eine Werkstückpalette gegriffen (*Robotload.Occupied* und *RobotGripper.Clamped*) haben. Der dazugehörige Bewegungsablauf (*Putting*), mit dem die Palette von oben in das Nullpunktspannsystem auf dem Maschinentisch gebracht werden kann, kann nur ausgeführt werden, wenn dieser frei (*MachiningTable.Empty*) und entspannt (*ClampingSystem.Unclamped*), und die Beladetür geöffnet (*SideDoor.Open*) ist. Dem entgegen steht die Anfahrbewegung zum Entladen der Maschine (*Approaching*), bei der der beladene Maschinentisch, bzw. die Palette darauf, mit einem leeren, geöffneten Robotergreifer angefahren werden muss. In beiden Fällen befindet sich der Roboter mit einer Werkstückpalette, die entweder von der Komponente ClampingSystem (CS) oder RobotGripper (RG) gespannt ist, an der Position *Table*. Zu diesem Zeitpunkt sind beide Komponenten MachiningTable (MT) oder RobotLoad (RL) belegt. Nach der Beladung muss sich der Roboter nun zurückziehen (*Retracting*), dabei muss die Maschine das Spannen der Palette übernehmen (*ClampingSystem.Clamped* und *RobotGripper.Unclamped*), und es erfolgt

die gegensätzliche Bewegung zu *Approaching*.

Tabelle 6.9: Abhängigkeiten von WorkpieceHandling (WH).

Von	**Nach**	**Skill**	**Bedingungen**
Outside	Putting	Put	RobotLoad.Occupied
Putting	Table		RobotGripper.Clamped
			SideDoor.Opened
			MachiningTable.Empty
			TransferPosition.Reached
			ClampingSystem.Unclamped
Outside	Approaching	Approach	RobotLoad.Empty
Approaching	Table		RobotGripper.Unclamped
			SideDoor.Opened
			MachiningTable.Occupied
			TransferPosition.Reached
			ClampingSystem.Clamped
Lifting	Outside		RobotGripper.Clamped
Table	Lifting	Lift	*ClampingSystem.Unclamped*
Retracting	Outside		RobotGripper.Unclamped
Table	Retracting	Retract	*ClampingSystem.Clamped*

Entsprechend gilt bei der Entladung der Maschine mit *Lifting* die gegensätzliche Bewegung zu *Putting*, und damit das Herausheben der Palette aus dem Spannsystem, unter den Bedingungen *ClampingSystem.Unclamped* und *RobotGripper.Clamped*. Damit lassen sich die Bedingungen für die Roboterbewegungen aus Sicht der Werkzeugmaschine modellieren.

6.4 Zusammenfassung

In diesem Kapitel wurde gezeigt, wie das Verhalten einer spezifischen Ausprägung einer Werkzeugmaschine anhand der einzelnen Zustandsautomaten ihrer Komponenten flexibel modelliert werden kann. Die Kombination aus Zustandsautomaten erlaubt es letztendlich, eine Werkzeugmaschine nicht nur mit ihren eigenen Komponenten, sondern auch im Kontext ihrer Anwendung zu beschreiben. Die hier modellierten einzelnen Zustandsautomaten resultieren in einem gemeinsamen Maschinenmodell[11] in SCXML, in dem alle Abhängigkeiten zwischen den internen Komponenten der Maschine, aber auch mit den externen Komponenten des Fertigungssystems, mit denen die Maschine kooperieren muss, abgebildet werden. In Bezug auf Hypothese 2 konnten damit nicht nur die Pragmatik innerhalb der Maschine, sondern auch der Zusammenhang zur Anwendung der Werkzeugmaschine in Kombination mit einem Handhabungsroboter modelliert werden. Semantik muss dabei die Grundlage zur Erkennung und Zuordnung der (aus Sicht der Werkzeugmaschine) externen Komponenten (WorkpieceHandling (WH), RobotGripper (RG) und RobotLoad (RL)) liefern, die der Werkzeugmaschine zwar konzeptuell bekannt sind, da sie für den maschinenspezifischen Handhabungsprozess eine Rolle spielen, aber den im tatsächlichen Fertigungssystem dafür vorhandenen Fähigkeiten erst zugeordnet werden müssen. Damit gibt dieses Kapitel in Kombination mit Unterabschnitt 5.2.3 einen Hinweis darauf, dass sich diese Modellierungsmethode ebenfalls für andere Fertigungszellen, bzw. Fertigungssysteme im Allgemeinen eignen könnte. Eine tatsächliche allgemeine Eignung muss jedoch zukünftig in weiteren Anwendungsszenarien überprüft werden.

[11] https://github.com/ttrau/graken/blob/thesis/spec/testmodel/cell-extended.scxml

Kapitel 7

Evaluierung

In diesem Kapitel erfolgt die Auswertung der vorher verwendeten Methode zur Modellierung der Pragmatik einer Werkzeugmaschine. Die Kombination an Zustandsautomaten aus Kapitel 6 wird dafür mit SCXML abgebildet. Es werden außerdem realistische Ausgangs- und Zielzustände der Zelle definiert, mit denen die Umsetzung bestimmter Aufgaben in der Zelle erreicht werden soll. Zu deren Evaluierung wird der SCXML Zustandsautomat zuerst in einen flachen, nicht-orthogonalen Zustandsautomaten umgewandelt, um die Suche durch den Graphen zu ermöglichen. Daraufhin wird eine BFS zwischen den Ausgangs- und Zielzuständen durchgeführt. Werden die jeweils kürzesten Pfade zwischen diesen Zuständen gefunden, muss evaluiert werden, ob diese tatsächlich einem geeigneten Steuerungsprozess zur Durchführung der vorgegebenen Aufgaben dienen.

7.1 Aufgaben, Ausgangszustände und Ziele

Die Aufgaben einer roboterisierten Fertigungszelle, welche die in Kapitel 6 modellierte Maschine enthält, können vielzählig sein. Als Hauptaufgabe dient hier die Fertigung mittels subtraktiver CNC-Bearbeitung von auf Paletten gespannten Werkstücken unter Berücksichtigung der automatisierten Werkstückhandhabung (Be- und Entladung der Maschine). Diese Aufgaben wurden folglich zur Evaluierung ausgewählt. Als grundlegender Ausgangszustand der Zelle kann in einer realen Umgebung die Kombination der aktuellen Einzelzustände der Maschinenkomponenten verwendet werden. Für die Evaluierung werden daher realistische Ausgangszustände in Tabelle 7.1 gewählt, in

denen sich die Zelle zum Zeitpunkt der Zuteilung einer bestimmten Aufgabe befinden könnte. Die Zielzustände enthalten letztendlich die in Zusammenhang mit der Aufgabe stehenden Zustände in den Einzelkomponenten der Maschine.

Tabelle 7.1: Ausgangszustände der Einzelkomponenten beim Beladen, Fertigen und Entladen.

Komponente	**Beladen**	**Fertigen**	**Entladen**
AuxiliaryDrives (AD)	Off	Off	On
ClampingSystem (CS)	Unclamped	Clamped	Clamped
EmergencyStop (ES)	Triggered	Normal	Normal
FrontDoor (FD)	Opened	Closed	Closed
HandlingKeyswitch (HK)	Off	On	On
MachiningTable (MT)	Empty	Occupied	Occupied
NumericalControl (NC)	Stopped	Stopped	Finished
OperationMode (OM)	Jog	Auto	Auto
RobotGripper (RG)	Clamped	Unclamped	Unclamped
RobotLoad (RL)	Occupied	Empty	Empty
SideDoor (SD)	Closed	Opened	Closed
TransferPosition (TP)	NotReached	Reached	Reached
WorkpieceHandling (WH)	Outside	Outside	Outside

7.1.1 Beladen

Das Ziel dieser Aufgabe ist es, die Fähigkeit zum *Beladen* der Zelle erfolgreich auszuführen. Diese entspricht einer *Pick & Place* Tätigkeit des Handhabungsroboters, bei welcher mit der Werkzeugmaschine interagiert werden muss, um ein vorher aufgenommenes Objekt in dieser abzulegen und zu spannen. In der ersten Spalte von Tabelle 7.1 sind die Ausgangszustände der Maschine vor der Beladung gewählt. Diese Zustände wurden absichtlich so gewählt, dass die Maschine noch nicht für eine automatisierte Handhabung bereit ist, da dafür grundlegende Voraussetzungen wie $EmergencyStop.Normal$, $HandlingKeyswitch.On$, $FrontDoor.Closed$ und $TransferPosition.Reached$ nicht gegeben sind. Eine korrekte Interpretation des Graphen zur Beladung der Maschine müsste die Notwendigkeit dieser Zustände bereits bei den

ersten Prozessschritten identifizieren. Weiterhin wird davon ausgegangen, dass der Maschinentisch leer ($MachiningTable.Empty$) ist und der Handhabungsroboter bereits eine zu beladende Palette aus seinem eigenen Arbeitsraum gegriffen hat ($Pick$). Der Teilaspekt dieser Aufgabe, ein Objekt abzulegen ($Place$), wird in zwei Subziele unterteilt, die dabei erreicht werden müssen: (1) *Putting* und (2) *Retracting*. Dazu ist es letztendlich notwendig, den Zustand $MachiningTable.Occupied$ bzw. $RobotLoad.Empty$ zu erreichen und in $WorkpieceHandling.Outside$ zurückzukommen, nachdem die Werkzeugmaschine mit einer Palette beladen wurde.

7.1.2 Fertigen

Die Ausgangszustände vor der Werkstückbearbeitung haben nach Tabelle 7.1 relativ wenige weitere Abhängigkeiten. Die Werkzeugmaschine ist beladen ($MachiningTable.Occupied$), der Roboter außerhalb der Maschine ($WorkpieceHandling.Outside$) und die Werkstückpalette gespannt ($ClampingSystem.Clamped$). Lediglich die Zustände $SideDoor.Closed$, $AuxiliaryDrives.On$ und $NumericalControl.Running$ sind noch zur Bearbeitung notwendig. Das Ziel dieser Aufgabe ist es den Zustand $NumericalControl.Finished$ herzustellen, bei der die Bearbeitung des Werkstücks erfolgreich abgeschlossen wurde.

7.1.3 Entladen

Die Entladung geht vom Zustand nach der Werkstückbearbeitung aus ($NumericalControl.Finished$). Das Ziel dieser Aufgabe für den Handhabungsroboter ist die Durchführung der gegensätzlichen Aktion zur Beladung. Unter Erreichung der Subziele $WorkpieceHandling.Approaching$ und $WorkpieceHandling.Lifting$ ($Pick$) sind die Zielzustände $MachiningTable.Empty$, $RobotLoad.Occupied$ und $WorkpieceHandling.Outside$ herzustellen.

7.2 Auflösung und Validierung der Zustandsautomaten

Die Auflösung der Hierarchie und Orthogonalität der Zustandsautomaten und deren anschließende Validierung erfolgt nach den in Unterabschnitt 5.2.4 beschriebenen Re-

geln. Diese wurden in einem Kommandozeilenprogramm graken[12,13] umgesetzt, mit dem SCXML Modelle zu einem einzelnen flachen Graphen verarbeitet und die Gültigkeit der Kanten anhand ihrer Bedingungen validiert werden können.

Tabelle 7.2: Ausführung von graken auf einem Intel® Core™ i7-10710U Prozessor.

Schritt	**Knoten**	**Kanten**	**Berechnungszeit**
Import	53	49	
Auflösung	373.249	6.034.176	00:00:19s
Validierung (intern + extern)	373.249	1.931.472	00:00:43s
Validierung (nur extern)	100.561	242.664	00:00:45s

In Tabelle 7.2 ist die Ausführung[14] des Programms für das in Kapitel 6 modellierte System[15] zusammengefasst. Die 53 Knoten beim Schritt **Import** sind darauf zurückzuführen, dass im SCXML Modell nicht nur die Summe der Zustände in den Komponenten (37 Knoten), sondern auch die 13 Komponenten selbst, ein initialer Pseudozustand des Gesamtsystems, die Werkzeugmaschine ($ManufacturingCell$) und ein zusammenfassender orthogonaler Zustandsautomat ($Systems$) als Knoten modelliert sind. Entsprechendes gilt für die 49 Kanten, die auch zwischen diesen zusätzlichen Knoten vorhanden sind, nicht nur zwischen den Einzelkomponenten.

Bei der **Auflösung** des orthogonalen Zustandsautomaten $Systems$ ergibt sich die Gesamtanzahl der neu entstehenden Knoten aus der Formel:

$$\begin{aligned} I &= |\mathbb{AD}| \times |\mathbb{CS}| \times |\mathbb{ES}| \times |\mathbb{FD}| \times |\mathbb{HK}| \times |\mathbb{MT}| \times |\mathbb{NC}| \\ &\quad \times |\mathbb{OM}| \times |\mathbb{RG}| \times |\mathbb{RL}| \times |\mathbb{SD}| \times |\mathbb{TP}| \times |\mathbb{WH}| \\ &= 2 \times 3 \times 2 \times 3 \times 2 \times 2 \times 4 \times 3 \times 3 \times 2 \times 3 \times 2 \times 6 \\ &= 373248 \end{aligned}$$

Bei der Auflösung der hierarchischen Komponenten $Systems$ und $ManufacturingCell$ werden diese beiden Knoten entfernt und der initiale Pseudozustand von $ManufacturingCell$ auf einen der neu entstandenen Knoten übertragen, mit dem sich eine Gesamtanzahl von **373.249 Knoten** ergibt.

[12] https://github.com/ttrau/graken/tree/ts

[13] Der Titel *graken* stammt von den Begriffen *graph* und *ken* (engl. für Wissen) und soll in Anlehnung an den Begriff *Kraken* das Wissen darstellen, dass in den vielen verzweigten Armen des Graphen hinterlegt ist.

[14] Ablauf: https://github.com/ttrau/graken/blob/ts/ts/graken.log

[15] https://github.com/ttrau/graken/blob/ts/spec/testmodel/cell-extended.scxml

Mit der Anschließenden **Validierung** der 6.034.176 neu entstandenen Zustandsübergänge, wird die Anzahl der Zustandsübergänge entweder auf die theoretisch im System (unter Berücksichtigung der internen Bedingungen) möglichen 1.931.472 oder nur die tatsächlich von extern veränderbaren 242.664 Zustandsübergänge[16] eingeschränkt. Bei der Validierung kann dabei unterschieden werden, ob eine Kante aufgrund ihrer Bedingungen (1) nie, (2) nur in Abhängigkeit von internen Bedingungen oder (3) immer passiert werden kann. Kanten der Kategorie (1) werden immer entfernt, Kategorie (3) dagegen immer beibehalten. Kategorie (2) kann unter Berücksichtigung der internen Bedingungen beibehalten werden. Der Unterschied liegt in der geplanten Anwendung. Soll der Graph einer möglichst ganzheitlichen Betrachtung unterzogen werden, bei der z. B. auch nicht komplett automatisiert lösbare Pfade gefunden werden sollen, ist die Berücksichtigung der internen Bedingungen dafür geeignet. Sollen dagegen nur vollständig automatisiert lösbare Pfade gefunden werden, ist es leichter, diese Kanten zu verwerfen. Damit wird die Graphsuche erleichtert, da keine zusätzlichen Regeln benötigt werden, um derartige Kanten auszuschließen. Grundsätzlich kann der letztendlich entstandene Graph auch Systemzustände enthalten, die in der Realität nie erreicht werden könnten, und hat damit Potential für eine weitergehende Reduktion.

7.3 Graphsuche und automatische Generierung von Steuerungsprozessen

Für die Graphsuche und Ableitung der Steuerungspfade wurde ein weiteres Kommandozeilenprogramm graken-eval[17], unter Verwendung der graph-tool python library [134], entwickelt. Mit diesem können die flachen, validierten Graphen importiert und, ausgehend von bestimmen Startzuständen des Gesamtsystems (entsprechend Tabelle 7.1), mittels BFS nach dem kürzesten Pfad zu einem Knoten durchsucht werden, der einem oder mehreren bestimmten Zuständen in den Anlagenkomponenten entspricht. Die gefundenen Pfade können in einem grafischen GRAFCET [135] Format ausgegeben werden, das die Prozessschritte, zugehörigen Aktionen und die notwendigen Übergangsbedingungen in Form einer Ablaufsteuerung darstellt. Die Graphsuche wurde entsprechend den in Abschnitt 7.1 definierten Ausgangszuständen, Zielen und Subzielen durchgeführt[18]. Die nachfolgenden GRAFCET Graphen wurden allesamt automatisch generiert und im Nachhinein nur zur besseren Druckbarkeit manuell in zwei Spalten aufgeteilt.

[16] Ablauf: https://github.com/ttrau/graken/blob/ts/ts/graken-extern.log
[17] https://github.com/ttrau/graken-eval/tree/ts
[18] Ablauf: https://github.com/ttrau/graken-eval/blob/ts/ts/script.log

7.3.1 Beladen

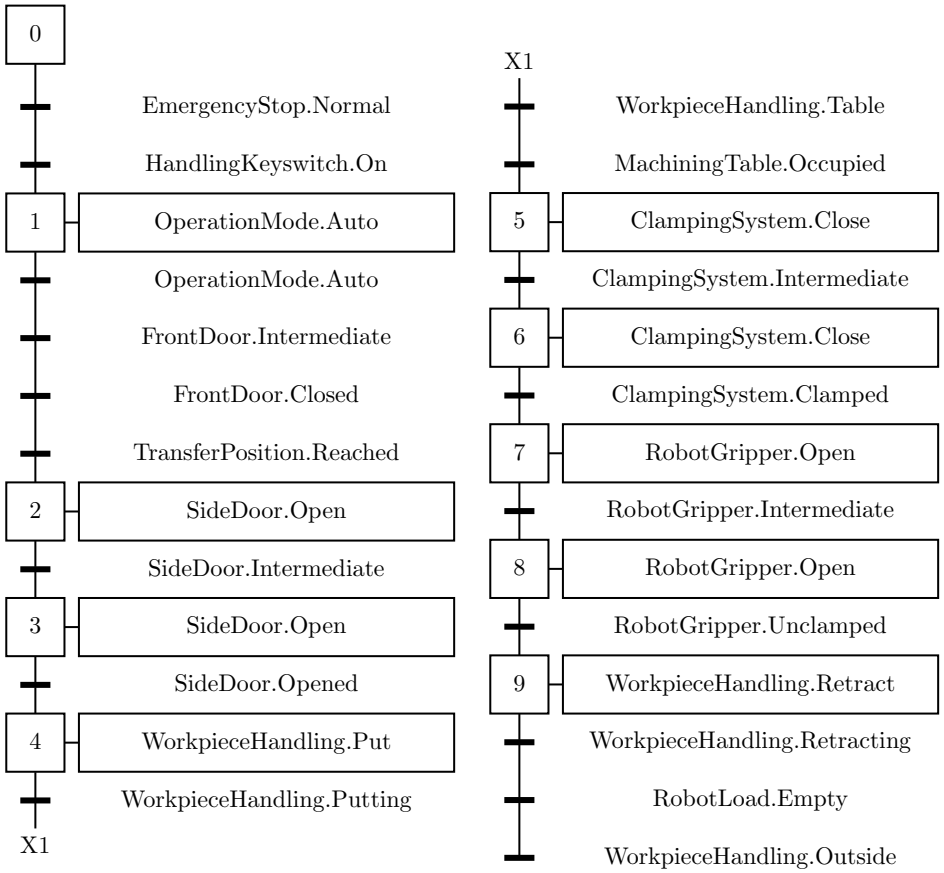


Abbildung 7.1: GRAFCET Steuerungsprozess zur Beladung.

Der zur Beladung der Werkzeugmaschine identifizierte Prozess ist in Abbildung 7.1 dargestellt und basiert auf den in Unterabschnitt 7.1.1 definierten Zielen und Subzielen. Auf Grund der Abhängigkeiten aller Komponenten von den Zuständen *EmergencyStop.Normal* und *HandlingKeyswitch.On*, entsprechend zu Abbildung 6.3, wurden diese Bedingungen direkt zu Beginn identifiziert. Da die Bedingungen aus Tabelle 6.9 für das erste Subziel *Workpiecehandling.Putting* vorschreiben, dass sich die Werkzeugmaschine in den Zuständen *SideDoor.Opened* und *TransferPosition.Reached* befindet, werden auch diese Abhängigkeiten früh identifiziert. Weil diese Signale rein interner Natur sind, und damit nicht von extern aktiv verändert werden können, kann nur deren Abhängigkeit zu bestimmten Zeitpunkten identifiziert werden, aber nicht wie diese Zustände hergestellt werden können. Möglich ist es aber trotzdem implizite Zusammenhänge dieser Signale mit anderen Zuständen, entsprechend Tabelle 6.7 oder Tabelle 6.8, zu modellieren. Diese weisen auf die sonstigen notwendigen Zustände hin, die für einen Zustandsübergang notwendig sind und unterstützen

damit die Identifikation korrekter Prozessschritte, haben aber keine weitere Ursache-Wirkungs-Beziehung. Solche könnten aber ebenfalls in Form von Abhängigkeiten, z. B. $RobotLoad.Occupied \wedge WorkpieceHandling.Table \rightarrow MachiningTable.Occupied$, modelliert werden. Beim Übergang vom Zustand $WorkpieceHandling.Putting$ zum nächsten Subziel $WorkpieceHandling.Retracting$ wird die Notwendigkeit vom Zustand $MachiningTable.Occupied$ derzeit nur inferiert, da dieser für das Subziel notwendig ist. Gleiches gilt für den Zustand $RobotLoad.Empty$, für welchen ebenfalls keine bekannte Ursache-Wirkungs-Beziehung existiert.

Auffällig sind beim Prozess in Abbildung 7.1 außerdem die jeweils doppelten Aktionen bei den Komponenten *SideDoor*, *ClampingSystem* und *RobotGripper* (Schritte 2, 3, 5, 6, 7 und 8). Diese sind auf die Art der Modellierung der Zustandsautomaten mit dem Zustand *Intermediate* in Abbildung 6.5, Abbildung 6.4 und Abbildung 6.15 zurückzuführen. Eine Modellierung dieser Zustandsautomaten könnte alternativ auch ohne die Trigger in den von *Intermediate* ausgehenden Zustandsübergängen umgesetzt werden, um die jeweils zweite Aktion zu ignorieren. Der Zustand *Intermediate* könnte ebenso durch die zwei Zustände *Opening* und *Closing* ersetzt werden, um eine genauere Auswertung zu ermöglichen. Da die Modellierung der Trigger in diesem Fall aber keinen Nachteil bei der Ausführung des Prozesses liefert und stattdessen einen Hinweis bietet, wie ein möglicher Fehlerzustand im Zustand *Intermediate* verlassen werden kann, wurde diese Art der Modellierung gewählt.

7.3.2 Fertigen

Die Prozessschritte zur Bearbeitung des vorher beladenen Werkstücks zum Erreichen des Ziels $NumericalControl.Finished$ sind in Abbildung 7.2 dargestellt. Dabei werden jedoch die Übertragung und Anwahl des dazu notwendigen Bearbeitungsprogramms vernachlässigt, da gewählte Programme zwar theoretisch als Zustände der Maschine, aber nicht in einem endlichen Zustandsautomaten darstellbar sind, weil eine unendliche Anzahl an unterschiedlichen Programmen existiert, die nicht im Vorhinein definierbar sind. Da das notwendige Programm aber sowieso abhängig vom Fertigungsauftrag ist, muss die Zellensteuerung die Übertragung und Anwahl auf der Werkzeugmaschine als Subziel des Fertigungsauftrags durchführen. Die Werkzeugmaschine sollte die dazu notwendigen Zustände in einem Zustandsautomaten darstellen, der angibt, wann ein Programm geladen oder (z. B. abhängig vom Zustand $NumericalControl.Ready$) zur Bearbeitung angewählt werden darf.

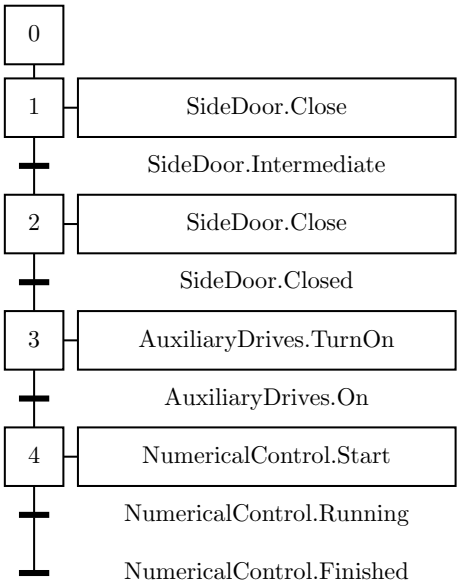


Abbildung 7.2: GRAFCET Steuerungsprozess zur Bearbeitung.

7.3.3 Entladen

Zuletzt sind auch die Prozessschritte zum Entladen der Werkzeugmaschine in Abbildung 7.3 dargestellt. Darin werden die Subziele $WorkpieceHandling.Approaching$ und $WorkpieceHandling.Lifting$ nacheinander erreicht, um das bearbeitete Werkstück aus der Werkzeugmaschine zu entladen und letztendlich die Zielzustände $MachiningTable.Empty$, $RobotLoad.Occupied$ und $WorkpieceHandling.Outside$ zu erreichen.

7.4 Zusammenfassung

Die in Kapitel 6 modellierte Pragmatik einer Fertigungszelle konnte, unter Anwendung der in Unterabschnitt 5.2.4 definierten Regeln zum Zusammenführen der orthogonalen Zustandsautomaten und einer BFS durch den daraus entstehenden Graph, verwendet werden, um die Steuerung des Gesamtsystems anhand seiner Ziele dynamisch abzuleiten. Damit konnte, in Hinblick auf Hypothese 3, in Abschnitt 7.2 gezeigt werden, dass (1) die Auflösung von orthogonalen und hierarchischen Graphen in einen einzelnen flachen Graphen möglich ist, und in Abschnitt 7.3, dass (2) daraus ziel-basiert maschinenspezifische Steuerungsprozesse abgeleitet werden können Damit konnte Hypothese 3 bestätigt werden. Die Kombination der pragmatischen und dynamischen Interopera-

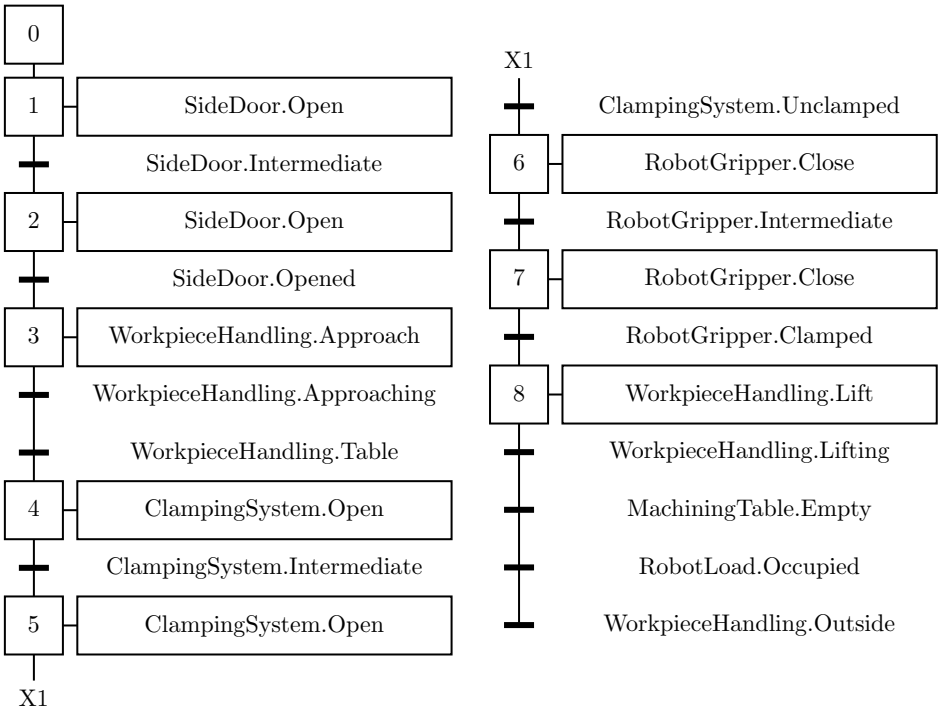


Abbildung 7.3: GRAFCET Steuerungsprozess zur Entladung.

bilität ermöglicht damit die ersten Schritte in Richtung einer agilen Automatisierung und Inbetriebnahme vom Fertigungssystemen. Nichtsdestotrotz müssen zukünftig weitere Untersuchungen durchgeführt werden, um auch eine allgemeine Anwendbarkeit dieser Methoden nachzuweisen.

Kapitel 8

Semiotik als Lösung zur agilen Automatisierung?

Agilität, in diesem Kontext die schnelle Anpassungsfähigkeit an Veränderungen, wurde in den letzten Jahrzehnten zu einem immer wichtigeren Thema, Auf europäischer Ebene soll nun durch Initiativen wie I4.0 besonders die Wettbewerbsfähigkeit gegenüber produzierenden Unternehmen aus dem europäischen Ausland langfristig gesichert werden, indem Internettechnologien und die damit einhergehende Standardisierung als Enabler höherer Agilität dienen. Teil dieser Entwicklung zur höheren Agilität ist es, Fertigungssysteme schneller an Marktänderungen anpassen zu können, also die Fähigkeit zur schnellen Inbetriebnahme, Automatisierung und Rekonfiguration solcher Systeme zu verbessern. Diese **Notwendigkeit zur agilen Automatisierung** ist die grundlegende Motivation dieser Arbeit und der damit verbundenen Forschungsfragen und Hypothesen.

8.1 Zusammenfassung

Zuerst wurden in **Kapitel 2** konventionelle Fertigungssysteme neueren Ansätzen gegenübergestellt, mit denen eine Verbesserung der Agilität erzielt werden könnte, und daraus die Grundlagen für **agil-rekonfigurierbare Fertigungssysteme** abgeleitet. Diese müssen folglich eine zumindest **teil-automatisierte und dynamische Komposition ihrer Fähigkeiten** ermöglichen, sind aber als komplexe Systeme oft strukturabhängig und besitzen viele Abhängigkeiten zwischen ihren Komponenten.

In **Kapitel 3** wurden zunächst unterschiedliche Modelle zur Interoperabilität untersucht und miteinander verglichen (u. a. LCIM, RAMI4.0, IIRA). Der Vergleich zeigt, dass viele der Modelle auf der Wissenspyramide oder der Semiotik aufsetzen und dabei die Interoperabilität technischer Systeme eher aus einer menschlichen Perspektive (Bionik) betrachten, während andere Modelle eine rein technische Sichtweise einnehmen und eher die Struktur, notwendigen Fähigkeiten und Eigenschaften technischer Systeme betrachten. Während die Standardisierung von Syntax und Semantik der Kommunikation zwischen den Maschinen dabei häufig als Grundlage für die Interoperabilität solcher Systeme gesehen wird, deuten die Wissenspyramide und die Semiotik mit der **Pragmatik** auf eine weitere erforderliche Komponente hin. Mit dieser wird, neben der durch die Semantik beschriebenen allgemeinen Bedeutung einer Sache, auch deren Bedeutung im Kontext einer Anwendung beschrieben. Für eine **dynamische Komposition der Fähigkeiten von komplexen Systemen** müssen deren Abhängigkeiten untereinander im Anwendungskontext berücksichtigt werden, wofür die Pragmatik einen wichtigen Beitrag leistet. Als Folge wurden unterschiedliche Arten der Modellierung von Semantik und Pragmatik untersucht, um einen geeigneten Lösungsansatz für die Komposition von Fertigungssystemen zur agilen Automatisierung zu identifizieren. Als mögliche Technologien wurden RDFS, OWL und OPC UA (für die Semantik), bzw. Petri Nets und verschiedene Ausprägungen von Zustandsautomaten in UML und SCXML (für die Pragmatik) identifiziert. Dabei fanden sich viele Kritiken aus der Zeit nach der Einführung des *Semantic Web* (2002 - 2010), dass semantische Beschreibungen hierfür alleine nicht ausreichen und ebenfalls Pragmatik notwendig sei.

Die daraus entstandenen Implikationen im Rahmen der Arbeit in **Kapitel 4** werfen drei zentrale Forschungsfragen auf. (F_1) *Inwiefern können die Behauptungen, dass Semantische Interoperabilität nicht ausreicht, unterstützt werden?* (F_2) *Wie kann das Wissen zur Anwendung von unterschiedlichen Geräten und/oder deren Komponenten so modelliert werden, dass dieses in einem komplexen Gesamtsystem verknüpft werden kann?* (F_3) *Wie können aus einem solchen Modell eines komplexen Systems Steuerungsprozesse für das Gesamtsystem abgeleitet werden?* Forschungsfrage 1 konnte mit Hypothese 0 bzw. Hypothese 1 in einem Gedankenexperiment zur Rolle der Semantik und anschließend in Unterabschnitt 4.4.3 überprüft werden. Daraus lässt sich schließen, dass semantische Interoperabilität alleine nicht ausreicht, um damit ein heterogenes, komplexes Systems so zu beschreiben, dass daraus eine Steuerungslogik abgeleitet werden kann.

Folglich mussten für die Semiotische Interoperabilität agiler Fertigungszellen in **Kapitel 5** Lösungsansätze für Forschungsfrage 2 und Forschungsfrage 3 gefunden und diese in einem gemeinsamen Konzept untergebracht werden. Das hierbei entwickelte Konzept basiert auf der Grundannahme, dass bereits von der Herstellerfirma Wissen

zur Anwendung von spezifischen Maschinen (auch im Kontext der Automatisierung von Fertigungszellen) in deren Steuerung und Betriebsanleitung einfließt. Dieses Wissen kann mit seiner Syntax, Semantik und Pragmatik (folglich seiner Semiotik) bereits bei der Entwicklung der Maschine so modelliert werden, dass es bei der Inbetriebnahme genutzt werden kann, um eine agile Automatisierung der Zelle umzusetzen. Zwar wurde im Konzept ein ganzheitlicher Ansatz, basierend auf dem LCIM, vorgestellt, um die für eine reale Umsetzung notwendigen Aspekte zu beschreiben, jedoch wurde in dieser Arbeit der Fokus auf die Modellierung der Pragmatik und die dynamische Ableitung der Steuerungsprozesse gelegt, da die übrigen Aspekte bereits umfassend erforscht sind.

Zur Modellierung der Pragmatik wurde im Konzept auf die Darstellung des Verhaltens der einzelnen Maschinenkomponenten mit Zustandsautomaten gesetzt. Die Aufteilung in einzelne Komponenten erlaubt nicht nur eine vereinfachte Sichtweise durch die Fokussierung auf Teilbereiche der Maschine, sondern auch eine flexible Erweiterbarkeit der Maschine mit weiteren Komponenten. Das gesamte Modell einer Maschine besteht dabei letztendlich aus einer Menge orthogonaler Zustandsautomaten, deren Abhängigkeiten untereinander als zusätzliche Regeln hinterlegt werden können. Um die Abhängigkeiten zwischen Maschinen im Kontext der Fertigungszelle abzubilden, können außerdem externe (ähnlich einem Platzhalter) Komponenten modelliert werden, die zur Laufzeit mit tatsächlichen Maschinen und deren Komponenten (z. B. über deren Semantik) verknüpft werden müssen. In **Kapitel 6** wurde Hypothese 2 in Hinblick auf die Modellierung des Verhaltens (1) einer Werkzeugmaschine selbst, und (2) der Werkzeugmaschine im Kontext der Automatisierung in einer Fertigungszelle überprüft. Auf Grund dieses induktiven Vorgehens durch Modellierung eines Spezialfalls kann für Forschungsfrage 2 geschlossen werden, dass sich diese Art der Modellierung wahrscheinlich auch für eine allgemeine Verwendung zur Modellierung des Verhaltens von Fertigungssystemen eignen könnte, hierfür aber noch weitere Forschungsarbeit notwendig ist.

Um Forschungsfrage 3 zu beantworten, erfolgte zuletzt in Kapitel 7 die Evaluierung der für die Dynamische Interoperabilität definierten Regeln (1) zur Umwandlung der orthogonalen Zustandsautomaten in einen einfachen, flachen Graphen und (2) zur Ableitung von Steuerungsprozessen aus diesem. Um Hypothese 3 zu überprüfen, wurden in Abschnitt 7.1 Aufgaben und Ziele definiert und das Modell der Werkzeugmaschine zuerst umgewandelt, dann mittels einer BFS automatisch nach geeigneten Steuerungsprozessen hinsichtlich dieser Aufgaben und den zugehörigen Zielzustände durchsucht. Das Auffinden geeigneter Steuerungsprozesse hat Hypothese 3 bestätigt und impliziert damit für Forschungsfrage 3, dass mit diesen Regeln auch das Ableiten von Steuerungsprozessen aus Modellen komplexer Systeme möglich ist.

8.2 Ausblick

In dieser Arbeit konnte letztendlich gezeigt werden, dass (1) semantische Beschreibungen von Maschinen alleine nicht ausreichen, um Steuerungslogik von heterogenen Maschinen in komplexen Systemen abzuleiten, (2) deren Pragmatik aber mit orthogonalen Zustandsautomaten der Einzelkomponenten, und Abhängigkeiten zwischen diesen, modelliert werden kann und daraus (3) maschinenspezifisch Steuerungsprozesse abgeleitet werden können. Folglich stellt sich in dieser Hinsicht, neben der Modellierung der Semantik, auch die Modellierung der Pragmatik als notwendig heraus. Die Beschreibung von Maschinen aus Sicht der Semiotik ist damit für heterogene, komplexe Systeme vielversprechender, als alleine die Syntax und Semantik zu modellieren und dabei auf eine statische Menge an standardisierten Verhaltensregeln, Steuerungsregeln und Anwendungen zu setzen. Die **semiotische Interoperabilität** könnte damit die Grundlage für die dynamische Komposition von Maschinen in komplexen Systemen bieten. Dafür stellen sich zumindest zwei notwendige Anforderungen, dass (1) das dafür notwendige Wissen in den Komponenten des Systems vorliegt und (2) dieses, falls es verteilt vorliegt, auch korrekt zusammengeführt und interpretiert werden kann.

Dahingehend ergeben sich verschiedenste Herausforderungen für die Weiterentwicklung und praktische Anwendbarkeit der in dieser Arbeit beschriebenen Methoden zur semiotischen Interoperabilität von Fertigungssystemen. Da durch diese ein Aufwand für die Herstellerfirma entsteht, der erst auf der Anwendungsseite einen wirklichen Nutzen bringt, kann eine tatsächliche Umsetzung hauptsächlich durch die Nachfrage danach getrieben, oder müsste über Forschungsmittel finanziert werden. Außerdem ist das Ziel einer autonomen Automatisierung der Komponenten noch in weiter Ferne, da man sich trotzdem an den Schnittstellen der Systeme auf eine einheitliche Semantik und zugrundeliegende Konzepte und Fähigkeiten einigen muss. Es ist daher unwahrscheinlich, dass sich in den nächsten Jahren schon viele Maschinen mit solchen Modellen selbst beschreiben. Ein erster Schritt wäre aber auch nicht zwangsläufig die Modellierung maschinen-übergreifender Abhängigkeiten, denn bereits die Steuerung einer einzelnen Maschine könnte auf Basis des internen Verhaltens der eigenen Komponenten inferiert werden. Damit würde bereits ein greifbarer Mehrwert geschaffen werden, indem den Unternehmen ein einfacherer Umgang mit neuen Maschinen ermöglicht wird. Einsatzgebiete reichen dabei von der Unterstützung einer teil-autonomen Automatisierung, über die Analyse und das Lösen von Fehlerzuständen, bis hin zur Unterstützung des Personals bei der Bedienung einer Maschine. Auch im Bereich von Modellierungssprachen und Modellierungstools ergeben sich Aufwände, aber auch Chancen. Eine Umsetzung in Form einer Begleitspezifikation für OPC UA könnte eine schnellere Verbreitung fördern und hätte dabei eine hohe Umsetzungswahrscheinlichkeit durch die graph-basierte Modellierung von OPC UA Adressräumen und bereits vorhandene In-

formationsmodelle für Zustandsautomaten. Außerdem würde es im Zusammenspiel mit anderen Spezifikationen die Fokussierung auf eine weniger strikte und flexiblere Standardisierung von Maschinenmodellen ermöglichen. Aus Sicht der Herstellerfirmen, die das Verhalten und die Verwendung der Anlage nun zusätzlich modellieren müssen, sind einfache Modellierungstools notwendig, die möglicherweise nicht nur unterstützen, sondern auch Regeln oder Programmcode aus einer SPS automatisch übersetzen können. Auch bei der Modellierungsmethodik selbst sind noch Erweiterungen möglich, die es erlauben, auch implizite Abhängigkeiten bzw. Effekte zu hinterlegen, mit denen umfangreichere Kausalketten gebildet werden können.

Literatur

[1] T. Trautner, I. Ayatollahi, D. Strutzenberger, T. Frühwirth, F. Pauker und B. Kittl, „Behavioral modeling of manufacturing skills in OPC UA for automated execution by an independent cell controller," in *Procedia CIRP*, 2020, In Press.

[2] B. J. Wallner, *Konzipierung und Umsetzung unterschiedlicher Steuerungsarchitekturen zur agilen Automatisierung von Fertigungszellen*. Wien, 2020. Adresse: https://repositum.tuwien.ac.at/urn:nbn:at:at-ubtuw:1-136183.

[3] F. Lang, *Prozessmodellierung für eine flexible Fertigungszelle*. Wien, 2020. Adresse: https://repositum.tuwien.ac.at/urn:nbn:at:at-ubtuw:1-135484.

[4] R. N. Nagel und R. Dove, „21st Century Manufacturing Enterprise Strategy: An Industry-lead View," Jg. 1, 1991.

[5] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy und H. Van Brussel, „Reconfigurable manufacturing systems," *CIRP Annals - Manufacturing Technology*, Jg. 48, Nr. 2, S. 527–540, 1999.

[6] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi und J. Ota, „Agile assembly system by 'Plug and Produce'," *CIRP Annals - Manufacturing Technology*, Jg. 49, Nr. 1, S. 1–4, 2000.

[7] L. Monostori, „Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, Jg. 17, S. 9–13, 2014. Adresse: http://dx.doi.org/10.1016/j.procir.2014.03.115.

[8] H. P. Wiendahl, H. A. ElMaraghy, P. Nyhuis, M. F. Zäh, H. H. Wiendahl, N. Duffie und M. Brieke, „Changeable Manufacturing - Classification, Design and Operation," *CIRP Annals - Manufacturing Technology*, Jg. 56, Nr. 2, S. 783–809, 2007.

[9] F. Jovane, Y. Koren und C. R. Boër, „Present and future of flexible automation: Towards new paradigms," *CIRP Annals - Manufacturing Technology*, Jg. 52, Nr. 2, S. 543–560, 2003. Adresse: https://linkinghub.elsevier.com/retrieve/pii/S0007850607632326.

[10] Y. Koren, *The Global Manufacturing Revolution*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2010.

[11] D. Klitou, J. Conrads, M. Rasmussen, L. Probst und B. Pedersen, „Digital Transformation Monitor Germany: Industrie 4.0," *European Commission Report*, Nr. January, 2017. Adresse: https://ec.europa.eu/growth/tools-databases/dem/monitor/sites/default/files/DTM_Industrie4.0.pdf.

[12] European Commission, „A Stronger European Industry for Growth and Economic Recovery," *COM*, Jg. 582, S. 1–33, 2012. Adresse: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2012:0582:FIN:EN:PDF.

[13] M. Blanchet, „Industrie 4.0: the new industrial revolution. How Europe will succeed," Nr. March, 2014.

[14] A. Roth, „Industrie 4.0 – Hype oder Revolution?" In *Einführung und Umsetzung von Industrie 4.0*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 1–15. Adresse: http://link.springer.com/10.1007/978-3-662-48505-7_1.

[15] H. Hirsch-Kreinsen und M. ten Hompel, „Digitalisierung industrieller Arbeit: Entwicklungsperspektiven und Gestaltungsansätze," in *Handbuch Industrie 4.0 Bd.3*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, S. 357–376. Adresse: http://link.springer.com/10.1007/978-3-662-53251-5_21.

[16] A. Roth und D. Siepmann, „Industrie 4.0 – Ausblick," in *Einführung und Umsetzung von Industrie 4.0*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 247–260. Adresse: http://link.springer.com/10.1007/978-3-662-48505-7_5.

[17] C. Brecher und M. Weck, *Werkzeugmaschinen Fertigungssysteme 1: Maschinenarten und Anwendungsbereiche*, Ser. VDI-Buch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, Bd. 9. Adresse: http://link.springer.com/10.1007/978-3-662-46565-3.

[18] T. Bauernhansl, „Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma," in *Handbuch Industrie 4.0 Bd.4*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, S. 1–31. Adresse: http://link.springer.com/10.1007/978-3-662-53254-6_1.

[19] C. Ellwein, A. Schmidt, A. Lechler und O. Riedel, „Distributed Manufacturing: A Vision about Shareconomy in the Manufacturing Industry," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, New York, NY, USA: ACM, 2019, S. 90–95. Adresse: https://dl.acm.org/doi/10.1145/3365265.3365270.

[20] Worldbank, *Manufacturing, value added (% of GDP) - European Union, Austria, Germany, France, Czech Republic*, 2020. Adresse: https://data.worldbank.org/indicator/NV.IND.MANF.ZS?end=2019&locations=EU-AT-DE-FR-CZ&start=2002&type=shaded&view=chart&year=2019 (besucht am 17. 07. 2020).

[21] M. Dastbaz, „Industry 4.0 (i4.0): The Hype, the Reality, and the Challenges Ahead," in *Industry 4.0 and Engineering for a Sustainable Future*, Bd. 0, Cham: Springer International Publishing, 2019, S. 1–11. Adresse: http://link.springer.com/10.1007/978-3-030-12953-8_1.

[22] T. A. Fechter, H. Jaich und C. Glockner, „Produktionsprozesse – Produkte fertigen und montieren," in *Maschinenbau*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, S. 1135–1159. Adresse: http://link.springer.com/10.1007/978-3-662-55882-9_32.

[23] Deutsches Institut für Normung (DIN) und Deutsche Kommission Elektrotechnik (DKE), „DIN and DKE Roadmap: German Standardization Roadmap Industrie 4.0," Techn. Ber., 2020, S. 1–77. Adresse: https://www.din.de/blob/65354/57218767bd6da1927b181b9f2a0d5b39/roadmap-i4-0-e-data.pdf.

[24] Institute of Electrical and Electronics Engineers (IEEE), *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, 1990.

[25] A. Zeid, S. Sundaram, M. Moghaddam, S. Kamarthi und T. Marion, „Interoperability in Smart Manufacturing: Research Challenges," *Machines*, Jg. 7, Nr. 2, S. 21, 2019. Adresse: https://www.mdpi.com/2075-1702/7/2/21.

[26] W. Mahnke, S. H. Leitner und M. Damm, *OPC unified architecture*. Berlin Heidelberg: Springer, 2009.

[27] OPC Foundation, *Joint Working Groups*, 2020. Adresse: https://opcfoundation.org/about/working-groups/joint-working-groups/ (besucht am 09. 11. 2020).

[28] P. Barnaghi, W. Wang, C. Henson und K. Taylor, „Semantics for the internet of things: Early progress and back to the future," *International Journal on Semantic Web and Information Systems*, Jg. 8, Nr. 1, S. 1–21, 2012.

[29] M. Schurig, C. Rabitsch und C. Ramsauer, „Agile Produktion: Ein Produktionskonzept für volatile Zeiten," *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, Jg. 109, Nr. 12, S. 956–959, 2014. Adresse: http://www.hanser-elibrary.com/doi/10.3139/104.111265.

[30] Dudenredaktion (o. J.), *„Agil" auf Duden online*. Adresse: https://www.duden.de/rechtschreibung/agil (besucht am 17. 07. 2020).

[31] W. Michel und R. Kraushaar, *Bausteine für die Fabrik der Zukunft: Eine Einführung in die rechnerintegrierte Produktion (CIM)*, L. Cronjäger, Hrsg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. Adresse: http://link.springer.com/10.1007/978-3-642-85192-6.

[32] C. Glockner, „Werkzeugmaschinen – Werkstücke mit formgebenden Werkzeugen bearbeiten," in *Maschinenbau*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, S. 1091–1133. Adresse: http://link.springer.com/10.1007/978-3-662-55882-9_31.

[33] J. Bauer, „Produktionscontrolling und -management mit SAP® ERP," *Produktionscontrolling und -management mit SAP® ERP*, 2012.

[34] Y. Y. Yusuf, M. Sarhadi und A. Gunasekaran, „Agile manufacturing: the drivers, concepts and attributes," *International Journal of Production Economics*, Jg. 62, S. 33–43, 1999.

[35] M. Berger, „Revolutionärer Einsatz Agiler Fertigungssysteme in der Großserienproduktion," in *Intelligent produzieren*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 21–31. Adresse: http://link.springer.com/10.1007/978-3-642-13101-1_1.

[36] C. Ellwein, O. Riedel, O. Meyer und D. Schel, „Rent'n'Produce: A Secure Cloud Manufacturing Platform for Small and Medium Enterprises," *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*, S. 1–6, 2018.

[37] H. A. ElMaraghy, „Flexible and reconfigurable manufacturing systems paradigms," *Flexible Services and Manufacturing Journal*, Jg. 17, Nr. 4 SPECIAL ISSUE, S. 261–276, 2006.

[38] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn und K. Ueda, „Cyber-physical systems in manufacturing," *CIRP Annals*, Jg. 65, Nr. 2, S. 621–641, 2016.

[39] A. Koestler, *The ghost in the machine.* Oxford, England: Macmillan, 1968, S. xvi, 384–xvi, 384.

[40] J. Jarvis, D. Jarvis, R. Rönnquist und L. C. Jain, „Holonic Manufacturing Systems," in *Studies in Computational Intelligence*, Bd. 106, 2008, S. 7–16. Adresse: http://link.springer.com/10.1007/978-3-540-77479-2_2.

[41] P. J. P. Leitao und D. F. Restivo, „An Agile and Adaptive Holonic Architecture for Manufacturing Control," *Faculty of Engineering of University of Porto*, Jg. Degree of, Nr. January, S. 297, 2004.

[42] H. Van Brussel, „Holonic Manufacturing Systems BT - CIRP Encyclopedia of Production Engineering," in, L. Laperrière und G. Reinhart, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, S. 654–659. Adresse: https://doi.org/10.1007/978-3-642-20617-7_6556.

[43] P. Leitão und F. Restivo, „A Holonic Control Approach for Distributed Manufacturing," *Knowledge and Technology Integration in Production and Services*, S. 263–270, 2002.

[44] J. Jasperneite, S. Hinrichsen und O. Niggemann, „„Plug-and-Produce“ für Fertigungssysteme,“ *Informatik-Spektrum*, Jg. 38, Nr. 3, S. 183–190, 2015. Adresse: http://link.springer.com/10.1007/s00287-015-0877-x.

[45] J. Pfrommer, D. Stogl, K. Aleksandrov, S. Escaida Navarro, B. Hein und J. Beyerer, „Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems,“ *At-Automatisierungstechnik*, Jg. 63, Nr. 10, S. 790–800, 2015.

[46] U. T. Bührer, C. Legat und B. Vogel-Heuser, „Changeability of manufacturing automation systems using an orchestration engine for programmable logic controllers,“ *IFAC-PapersOnLine*, Jg. 28, Nr. 3, S. 1573–1579, 2015.

[47] S. Henning, O. Niggemann, J. Otto und S. Schriegel, „A descriptive engineering approach for cyber-physical systems,“ *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, Nr. March 2015, S. 7–11, 2014.

[48] J. Walter, K. Grüttner und W. Nebel, „Using IEC 61499 and OPC-UA to implement a self-organising plug and produce system,“ *CEUR Workshop Proceedings*, Jg. 2245, S. 475–484, 2018.

[49] M. Naumann, K. Wegener und R. D. Schraft, „Control architecture for robot cells to enable Plug'n'Produce,“ in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007, S. 287–292.

[50] D. Siepmann und N. Graef, „Industrie 4.0 – Grundlagen und Gesamtzusammenhang,“ in *Einführung und Umsetzung von Industrie 4.0*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 17–82. Adresse: http://link.springer.com/10.1007/978-3-662-48505-7_2.

[51] „Integrierte Forschungsagenda Cyber-Physical Systems,“ *acatech Studie*, Nr. März, E. M. Geisberger und M. Broy, Hrsg., S. 1–297, 2012.

[52] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan und P. Friess, „Internet of Things IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps,“ Techn. Ber., 2015.

[53] Plattform Industrie 4.0, *Glossary*, 2020. Adresse: https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/Glossar/glossar.html (besucht am 08. 09. 2020).

[54] K. Fuchs-Kittowski, *Wissens-Ko-Produktion - Organisationsinformatik*. 2001.

[55] A. Aamodt und M. Nygård, „Different roles and mutual dependencies of data, information and knowledge,“ *Data & Knowledge Engineering*, Jg. 16, Nr. 3, S. 191–222, 1995.

[56] C. W. Morris, *Foundations of the Theory of Signs*. University of Chicago Press Cambridge University Press, 1938.

[57] A. Tolk und J. Muguira, „The Levels of Conceptual Interoperability Model," *Fall Simulation Interoperability Workshop*, Nr. September, S. 1–9, 2003.

[58] C. D. Turnitsa, „Extending the levels of conceptual interoperability model," in *Proceedings IEEE summer computer simulation conference, IEEE CS Press*, 2005.

[59] A. Tolk, C. D. Turnitsa und S. Y. Diallo, „Ontological implications of the levels of conceptual interoperability model," *WMSCI 2006 - The 10th World Multi-Conference on Systemics, Cybernetics and Informatics, Jointly with the 12th International Conference on Information Systems Analysis and Synthesis, ISAS 2006 - Proc.*, Jg. 4, S. 105–111, 2006.

[60] W. Wenguang, A. Tolk und W. Wang, „The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S," *Proceedings of Spring Simulation Multiconference (SpringSim'09). San Diego, CA, USA*, 2009. Adresse: http://arxiv.org/abs/0908.0191.

[61] H. Van Der Veer und A. Wiles, „Achieving Technical Interoperability: the ETSI Approach," *European Telecommunications Standards Institute*, Nr. 3, S. 29, 2008. Adresse: https://portal.etsi.org/CTI/Downloads/ETSIApproach/IOPwhitepaperEdition3final.pdf.

[62] W. Li, K. Liu und S. Liu, „Semiotic interoperability a critical step towards systems integration," *IC3K 2013; KDIR 2013 - 5th International Conference on Knowledge Discovery and Information Retrieval and KMIS 2013 - 5th International Conference on Knowledge Management and Information Sharing, Proc.*, S. 508–513, 2013.

[63] ——, „Semiotics in Interoperation for Information Systems Working Collaboratively," in *Communications in Computer and Information Science*, Bd. 454, 2015, S. 370–386. Adresse: http://link.springer.com/10.1007/978-3-662-46549-3_24.

[64] R. Stamper, *Information in Business and Administrative Systems*. London; New York: Wiley, 1973.

[65] *ISO/IEC 21823-1:2019-02, Internet of things (IoT) - Interoperability for internet of things systems - Part 1: Framework*, 2019.

[66] C. H. Asuncion und M. J. van Sinderen, „Pragmatic interoperability: A systematic review of published definitions," *IFIP Advances in Information and Communication Technology*, Jg. 326, S. 164–175, 2010.

[67] C. H. Asuncion und M. van Sinderen, „Towards Pragmatic Interoperability in the New Enterprise — A Survey of Approaches," in *Lecture Notes in Business Information Processing*, Bd. 76 LNBIP, 2011, S. 132–145. Adresse: http://link.springer.com/10.1007/978-3-642-19680-5_12.

[68] „DIN SPEC 91345:2016-04," Nr. April, 2016.

[69] P. Adolphs und U. Epple, „Statusreport: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)," *VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik*, Nr. April, S. 1–32, 2015. Adresse: https://www.vdi.de/fileadmin/user_upload/VDI-GMA_Statusreport_Referenzarchitekturmodell-Industrie40.pdf.

[70] P. Adolphs, A. Sören, H. Bedenbender und M. Billmann, „Statusreport: Fortentwicklung des Referenzmodells für die Industrie 4.0 - Komponente," *VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik*, Nr. April, S. 1–48, 2016. Adresse: https://www.vdi.de/fileadmin/vdi_de/redakteur_dateien/gma_dateien/6146_PUB_GMA_ZVEI_Statusreport_-_RAMI_4-0_Struktur_der_Verwaltungsschale_Internet.pdf.

[71] Plattform Industrie 4.0, „Verwaltungsschale in der Praxis - Wie definiere ich Teilmodelle , beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen," 2019.

[72] H. Bedenbender, A. Bentkus, U. Epple, T. Hadlich, M. Hankel, R. Heidel, O. Hillermeier, M. Hoffmeister und H. Huhle, „Relationships between I4.0 Components – Composite Components and Smart Production," *Plattform Industrie 4.0*, S. 56, 2017. Adresse: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/hm-2018-relationship.html.

[73] S. Bader, E. Barnstedt, H. Bedenbender, M. Billman, B. Boss und A. Braunmandl, „Details of the Asset Administration Shell Part 1 - The exchange of information between partners in the value chain of Industrie 4.0," *Plattform Industrie 4.0*, S. 473, 2020. Adresse: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html.

[74] „VDI/VDE 2193-1:2020-04, Language for I4.0 components - Part 1: Structure of messages," 2020.

[75] „VDI/VDE 2193-2:2020-01, Language for I4.0 components - Part 2: Interaction protocol for bidding procedures," 2019.

[76] J. Reich, „Eine semantische Klassifikation von Systeminteraktionen," in *INFORMATIK 2015*, D. W. Cunningham, P. Hofstedt, K. Meer und I. Schmitt, Hrsg., Bonn: Gesellschaft für Informatik e.V., 2015, S. 1545–1559. Adresse: https://dl.gi.de/bitstream/handle/20.500.12116/2140/1545.pdf.

[77] A. Neubacher, A. Kraft, A. Willner, J. Reich, J. Wende, M. Lieske, R. Steinke, S. Grieß und T. Usländer, „Vorschlag zur systematischen Klassifikation von Interaktionen in Industrie 4.0 Systemen," Bitkom, Hrsg., S. 1–50, 2020. Adresse: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Bitkom_Thesenpapier.html.

[78] C. Diedrich, A. Bieliaiev, J. Bock, A. Gössling, R. Hänisch, A. Kraft, F. Pethig, O. Niggemann, J. Reich, F. Vollmar und J. Wende, „Interaktionsmodell für Industrie 4.0 Komponenten," *at - Automatisierungstechnik*, Jg. 65, Nr. 1, S. 5–18, 2017. Adresse: https://www.degruyter.com/doi/10.1515/auto-2016-0118.

[79] „The Industrial Internet of Things Volume G1: Reference Architecture," 2019. Adresse: https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf.

[80] S.-W. Lin, B. Murphy, E. Clauser, U. Loewen, R. Neubert, G. Bachmann, M. Pai und M. Hankel, „Architecture Alignment and Interoperability: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper," *Plattform Industrie 4.0*, S. 19, 2017. Adresse: https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf.

[81] D. Weidemann und R. Drath, „Bewertung und Ausblick," in *Datenaustausch in der Anlagenplanung mit AutomationML*, R. Draht, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 307–319. Adresse: http://link.springer.com/10.1007/978-3-642-04674-2_7.

[82] T. Berners-Lee, J. Hendler und O. Lassila, „The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, Jg. 284, Nr. 5, S. 1–5, 2001.

[83] D. L. McGuinness und F. van Harmelen, *OWL Web Ontology Language Overview. Technical report, W3C.* 2004. Adresse: http://www.w3.org/TR/owl-features/ (besucht am 22. 07. 2020).

[84] T. R. Gruber, „A translation approach to portable ontology specifications," *Knowledge Acquisition*, Jg. 5, Nr. 2, S. 199–220, 1993. Adresse: https://linkinghub.elsevier.com/retrieve/pii/S1042814383710083.

[85] D. McComb, „Terms: Vocabulary, Taxonomy, and Ontology," in *Semantics in Business Systems*, Elsevier, 2003, S. 41–64. Adresse: https://linkinghub.elsevier.com/retrieve/pii/B9781558609174500064.

[86] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll und B. McBride, *RDF 1.1 Concepts and Abstract Syntax*, 2014. Adresse: https://www.w3.org/TR/rdf11-concepts/ (besucht am 22. 07. 2020).

[87] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler und F. Yergeau, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 2008. Adresse: https://www.w3.org/TR/xml/ (besucht am 23. 07. 2020).

[88] D. Brickley, R. Guha und B. McBride, *RDF Schema 1.1*, 2014. Adresse: https://www.w3.org/TR/rdf-schema/ (besucht am 22. 07. 2020).

[89] OPC Foundation, *Unified Architecture*, 2020. Adresse: https://opcfoundation.org/about/opc-technologies/opc-ua/ (besucht am 27. 07. 2020).

[90] M. Graube, S. Hensel, C. Iatrou und L. Urbas, „Information models in OPC UA and their advantages and disadvantages," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, S. 1–8, 2017.

[91] A. Perzylo, S. Profanter, M. Rickert und A. Knoll, „OPC UA NodeSet Ontologies as a Pillar of Representing Semantic Digital Twins of Manufacturing Resources," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Bd. 2019-Septe, IEEE, 2019, S. 1085–1092. Adresse: https://ieeexplore.ieee.org/document/8868954/.

[92] S. Profanter, A. Breitkreuz, M. Rickert und A. Knoll, „A Hardware-Agnostic OPC UA Skill Model for Robot Manipulators and Tools," in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Bd. 2019-Septe, IEEE, 2019, S. 1061–1068. Adresse: https://ieeexplore.ieee.org/document/8869205/.

[93] F. W. Neiva, J. M. N. David, R. Braga und F. Campos, „Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature," *Information and Software Technology*, Jg. 72, S. 137–150, 2016. Adresse: https://linkinghub.elsevier.com/retrieve/pii/S0950584916000021.

[94] M. Bravo und J. Velazquez, „Discovering Pragmatic Similarity Relations between Agent Interaction Protocols," *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, S. 128–137, 2008.

[95] A. Tolk, „What Comes After the Semantic Web - PADS Implications for the Dynamic Web," in *20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*, IEEE, 2006, S. 55–55. Adresse: http://ieeexplore.ieee.org/document/1630709/.

[96] S. Liu, W. Li und K. Liu, „Assessing Pragmatic Interoperability of Information Systems from a Semiotic Perspective," in *Service Science and Knowledge Innovation*, K. Liu, S. R. Gulliver, W. Li und C. Yu, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, S. 32–41. Adresse: http://link.springer.com/10.1007/978-3-642-55355-4_4.

[97] A. de Moor, M. Keeler und G. Richmond, „Towards a Pragmatic Web," in *Lecture Notes in Computer Science*, Ser. Lecture Notes in Computer Science April, U. Priss, D. Corbett und G. Angelova, Hrsg., Bd. 2393, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 235–249. Adresse: http://link.springer.com/10.1007/3-540-45483-7_18.

[98] A. de Moor, „Patterns for the Pragmatic Web,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, June, Bd. 3596 LNAI, 2005, S. 1–18. Adresse: http://link.springer.com/10.1007/11524564_1.

[99] M. P. Singh, „The pragmatic web,“ *IEEE Internet Computing*, Jg. 6, Nr. 3, S. 4–5, 2002. Adresse: http://ieeexplore.ieee.org/document/1003124/.

[100] M. Schoop, A. De Moor und J. L. Dietz, „The pragmatic Web: A manifesto,“ *Communications of the ACM*, Jg. 49, Nr. 5, S. 75–76, 2006.

[101] M. P. Singh, „The Pragmatic Web: Preliminary Thoughts,“ 2002.

[102] C. A. Petri, „Kommunikation mit Automaten,“ 1962.

[103] A. M. Turing, „On Computable Numbers, with an Application to the Entscheidungsproblem,“ *Proceedings of the London Mathematical Society*, Jg. s2-42, Nr. 1, S. 230–265, 1937. Adresse: http://doi.wiley.com/10.1112/plms/s2-42.1.230.

[104] G. H. Mealy, „A Method for Synthesizing Sequential Circuits,“ *Bell System Technical Journal*, Jg. 34, Nr. 5, S. 1045–1079, 1955.

[105] E. F. Moore, „Gedanken-Experiments on Sequential Machines,“ 1956.

[106] J. Hopcroft, R. Motwani und J. Ullman, *Introduction to Automata Theory , Languages , and Computation, 3rd edition*, 3. Aufl. Pearson, 2006, S. 535.

[107] V. Alagar und K. Periyasamy, „Automata,“ in *Specification ofSoftware Systems*, London: Springer London, 2011, S. 77–103. Adresse: http://link.springer.com/10.1007/978-0-85729-277-3_6.

[108] D. Harel, „Statecharts: a visual complex systems*,“ *Science of Computer Programming*, Jg. 8, Nr. 3, S. 231–274, 1987.

[109] D. Harel und A. Naamad, „The STATEMATE Semantics of Statecharts,“ *ACM Transactions on Software Engineering and Methodology*, Jg. 5, Nr. 4, S. 293–333, 1996.

[110] E. M. Clarke, W. Klieber, M. Nováček und P. Zuliani, „Model Checking and the State Explosion Problem,“ in *Tools for Practical Software Verification: LASER, International Summer School 2011, Elba Island, Italy, Revised Tutorial Lectures*, 2005, B. Meyer und M. Nordio, Hrsg., Bd. 1041377, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 1–30. Adresse: http://link.springer.com/10.1007/978-3-642-35746-6_1.

[111] B. Rumpe, *Modeling with UML*. Cham: Springer International Publishing, 2016. Adresse: http://link.springer.com/10.1007/978-3-319-33933-7.

[112] Object Management Group, *OMG Unified Modeling Language – Version 2.5.1*, 2017. Adresse: https://www.omg.org/spec/UML/2.5.1.

[113] ——, *An OMG Systems Modeling Language – Version 1.6*, 2019. Adresse: http://www.omg.org/spec/SysML/20161101.

[114] L. Berardinelli, A. Mazak, O. Alt, M. Wimmer und G. Kappel, „Model-Driven Systems Engineering: Principles and Application in the CPPS Domain," in *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, Cham: Springer International Publishing, 2017, S. 261–299. Adresse: http://link.springer.com/10.1007/978-3-319-56345-9_11.

[115] J. Barnett, R. Akolkar, R. J. Auburn, M. Bodell, D. C. Burnett, J. Carter, S. McGlashan, T. Lager, M. Helbing und R. Hosn, *State Chart XML (SCXML): State machine notation for control abstraction*, 2015. Adresse: http://www.w3.org/TR/scxml/ (besucht am 26. 08. 2020).

[116] R. Auburn, P. Baggia und M. Scott, *Voice Browser Call Control: CCXML Version 1.0*, 2011. Adresse: https://www.w3.org/TR/ccxml/ (besucht am 26. 08. 2020).

[117] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski und P. Zanini, „Skill-based Engineering and Control on Field-Device-Level with OPC UA," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Jg. 2019-Septe, Nr. September, S. 1101–1108, 2019.

[118] S. Malakuti, P. Zimmermann, J. Grothoff, C. Wagner, J. Bock, M. Weser, P. Venet, M. Wiegand und A. Bayha, „Challenges in Skill-based Engineering of Industrial Automation Systems*," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2018, S. 67–74. Adresse: https://ieeexplore.ieee.org/document/8502635/.

[119] A. F. Cutting-Decelle, R. I. Young, J. J. Michel, R. Grangel, J. Le Cardinal und J. P. Bourey, „ISO 15531 MANDATE: A product-process-resource based approach for managing modularity in production management," *Concurrent Engineering Research and Applications*, Jg. 15, Nr. 2, S. 217–235, 2007.

[120] A. Björkelund, J. Malec, K. Nilsson, P. Nugues und H. Bruyninckx, „Knowledge for intelligent industrial robots," *AAAI Spring Symposium - Technical Report*, Jg. SS-12-02, Nr. Schmolze 1986, S. 7–12, 2012.

[121] A. Perzylo, J. Grothoff, L. Lucio, M. Weser, S. Malakuti, P. Venet, V. Aravantinos und T. Deppe, „Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective," *Proceedings of the IFAC Conference on Manufacturing Modeling, Management, and Control (MIM)*, Nr. December, S. 1682–1688, 2019.

[122] H. Van Brussel und P. Valckenaers, „Design of Holonic Manufacturing Systems," *Journal of Machine Engineering*, Jg. 17, Nr. 3, S. 5–23, 2017.

[123] F. Blackler, „Knowledge, Knowledge Work and Organizations: An Overview and Interpretation," *Organization Studies*, Jg. 16, Nr. 6, S. 1021–1046, 1995.

[124] „VDMA 34180:2016-03, Beschreibung der Schnittstellen zwischen Automation und Maschine," 2016.

[125] *MTConnect Part 1.0: Overview and Fundamentals - Version 1.6.0*, 2020. Adresse: https://docs.mtconnect.org/MTC_Part_1-0-Overview_and_Fundamentals_1-6-0.pdf.

[126] „MTConnect Part 2.0: Devices Information Model - Version 1.6.0," MTConnect Institute, Techn. Ber., 2020. Adresse: https://docs.mtconnect.org/MTC_Part_2-0_Devices_Information_Model_1-6-0.pdf.

[127] *MTConnect Part 4.0: Assets Information Model - Version 1.6.0*, 2020. Adresse: https://docs.mtconnect.org/MTC_Part_4-0_Assets_Information_Model_1-6-0.pdf.

[128] „MTConnect Part 4.1: Cutting Tools - Version 1.6.0," MTConnect Institute, Techn. Ber., 2020. Adresse: https://docs.mtconnect.org/MTC_Part_4-1_Cutting_Tools_1-6-0.pdf.

[129] *MTConnect Part 5: Interfaces - Version 1.6.0*, 2020. Adresse: https://docs.mtconnect.org/MTC_Part_5-0_Interfaces_1-6-0.pdf.

[130] K. Erciyes, „The Computational Model," in, 2013, S. 23–37. Adresse: http://link.springer.com/10.1007/978-1-4471-5173-9_3.

[131] U. Cheramangalath, R. Nasre, Y. N. Srikant, U. Cheramangalath, R. Nasre und Y. N. Srikant, „Graph Algorithms and Applications," *Distributed Graph Analytics*, S. 31–60, 2020.

[132] K. Erciyes, „Graphs," in *The Jerusalem Talmud,First order: Zeraim, Tractate Berakhot*, Bd. 3, 2013, S. 11–21. Adresse: http://link.springer.com/10.1007/978-1-4471-5173-9_2.

[133] U. Cheramangalath, R. Nasre und Y. N. Srikant, „Introduction to Graph Analytics," in *Distributed Graph Analytics*, Cham: Springer International Publishing, 2020, S. 1–29. Adresse: http://link.springer.com/10.1007/978-3-030-41886-1_1.

[134] T. P. Peixoto, „The graph-tool python library," *figshare*, 2014. Adresse: http://figshare.com/articles/graph_tool/1164194.

[135] „DIN EN 60848:2014-12 - GRAFCET, Spezifikationssprache für Funktionspläne der Ablaufsteuerung," 2014.